Diagnose elektronischer Fahrzeugsysteme durch Strukturanalysen

Von der Fakultät für Elektrotechnik und Informationstechnik

der Technischen Universität Carolo-Wilhelmina

zu Braunschweig

zur Erlangung der Würde

eines Doktor-Ingenieurs (Dr.-Ing.)

genehmigte

Dissertation

von

Dipl.-Ing. Mirko Harms

aus Wolfenbüttel

Eingereicht am:	10.05.2007
Mündliche Prüfung am:	11.06.2007
Berichterstatter:	Prof. Dr.-Ing. Jörn-Uwe Varchmin
	Prof. Dr.-Ing. Dr. h.c. Eckehard Schnieder
Vorsitz:	Prof. Dr.-Ing. Ulrich Seiffert

Braunschweig, 2007

Veröffentlichungen über den Inhalt der Arbeit sind
nur mit schriftlicher Genehmigung der Volkswagen AG zugelassen.

Vorwort

Die vorliegende Dissertation entstand im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Elektrische Messtechnik und Grundlagen der Elektrotechnik an der Technischen Universität Braunschweig.

Ganz besonderer Dank gilt Prof. Dr.-Ing. Jörn-Uwe Varchmin für die Betreuung dieser Arbeit. Er hat sie mit seiner Unterstützung in fachlicher, aber auch außerfachlicher Hinsicht erst zu einem realen Werk werden lassen.

Prof. Dr.-Ing. Eckehard Schnieder danke ich für die Übernahme der Mitberichterstattung und für die hilfreichen Diskussionen und Anregungen. Ebenso danke ich Herrn Prof. Dr.-Ing. Ulrich Seiffert für die freundliche Leitung des Promotionsverfahrens.

Herrn Andreas Breuer danke ich für die zahlreichen offenen Diskussionen während der Entstehung dieser Arbeit, diese haben wesentlich zu ihrem erfolgreichen Abschluss beigetragen. Gleiches gilt auch für alle anderen Kollegen und mitwirkenden Studenten, wobei ich den Herren Dr.-Ing. Marc Horstmann, Dr.-Ing. Martin Mutz und Bastian Florentz aufgrund der besonders intensiven Zusammenarbeit im Projekt explizit danken möchte.

Dankbar bin ich ebenso Herrn Dr.-Ing. Michael Schmidt für das Lektorat der Arbeit und die immer konstruktiven Anregungen während der praktischen Arbeit.

Abschließend gilt mein Dank meiner Familie, insbesondere meiner Frau Mirna Harms für die kontinuierliche und geduldige Unterstützung des gesamten Vorhabens.

Inhalt

Kurzzeichen und Begriffe

ADAC	Allgemeiner Deutscher Automobil-Club
ASAM	Association for Standardisation of Automation and Measurement
ATMS	Assumption based Truth Management System
AUTOSAR	Automotive Open System Architecture
CAD	Computer-Aided Design
CAE	Computer Aided Engineering
CAN	Controller Area Network
CASE	Computer-Aided System Engineering
CON	Connector (Stecker)
CS	Chip Select
CTRL	Control (Steuerleitung)
DTC	Diagnostic Trouble Code
EMV	Elektromagnetische Verträglichkeit
EOBD	European On-Board Diagnosis
ESP	Elektronisches Stabilitäts Programm
FMEA	Fehlermöglichkeits- und Einfluss-Analyse
GAL	Gate Array Logic
GDE	General Diagnosis Engine
IO	Input/Output
IP	Intellectual Property
KWP	Keyword Protocol
LIN	Local Interconnect Network
MCD	Measurement, Calibrations, Diagnostics
MOST	Media Oriented Systems Transport
OBD	On-Board Diagnosis

OEM	Original Equipment Manufacturer
OSEK	Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug
P	Port
PWR	Power
RAM	Random Access Memory
RE	Requirements Engineering
ROM	Read-Only Memory
RTC	Real Time Clock
SRAM	Static Random Access Memory
STEP-X	Strukturierter Entwicklungsprozess für automotive Anwendungen
TP	Transport Protocol
UART	Universal Asynchronous Receiver/ Transmitter
UML	Unified Modeling Language
VDI	Verein Deutscher Ingenieure
VDX	Vehicle Distributed eXecutive
XMI	XML Metadata Interchange
XML	Extensible Markup Language
ZVB	Zentrum für Verkehr Braunschweig

1. Einleitung und Motivation

1.1. Problemstellung

In heutigen Fahrzeugsystemen ist ein großer Anteil der Funktionen auf den Wunsch nach hoher Sicherheit, großem Komfort und der Reduktion der Schadstoffemissionen zurückzuführen. Dies wird durch den Einsatz von immer mehr Elektrik und Elektronik realisiert. Das zeigt sich auch anhand des prognostizierten weltweiten Umsatzes in diesem Bereich, der sich zwischen 2002 und 2015 mehr als verdoppeln wird. Vornehmlichen Anteil an dieser Entwicklung hat danach die IC- und die Software-Entwicklung mit jährlichen Zuwachsraten zwischen 10% und 13% [Merc04].

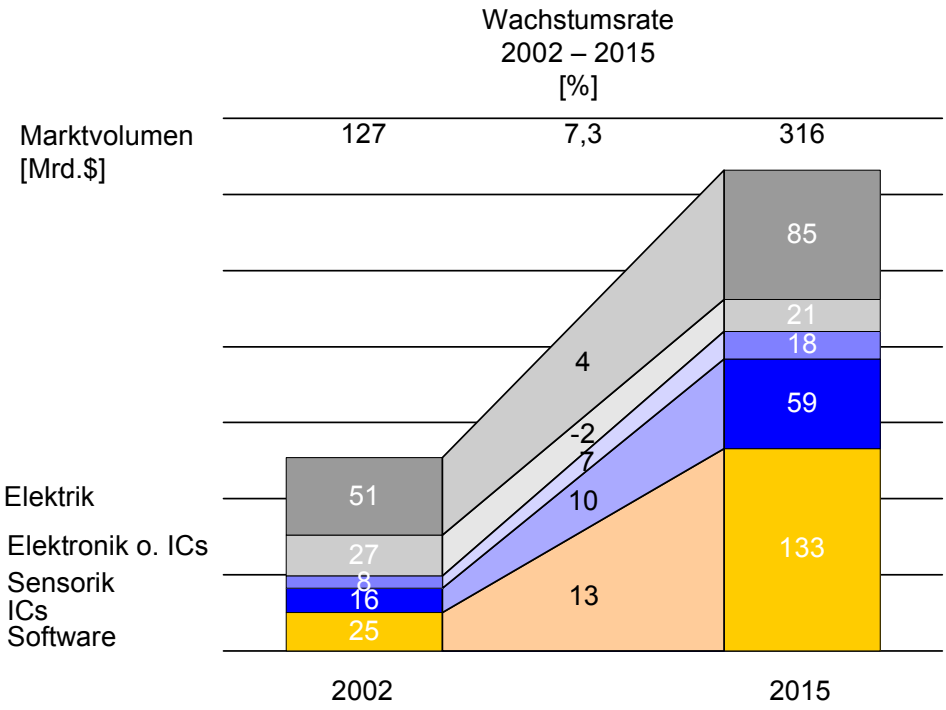


Abbildung 1-1 Prognose der globalen Umsätze für Elektrik/Elektronik im Automobil [Merc04]

Es zeigt sich auch anhand des Anteils der Pannen, die der ADAC in seiner jährlichen Pannenstatistik der Elektrik und Elektronik zuschreibt, dass das elektrische System immer mehr Verantwortung am Liegenbleiben von Kraftfahrzeugen hat und die steigende Komplexität dieser Systeme nicht geeignet beherrscht wird. Auch wenn die vordergründige Betrachtung ergibt, dass sich diese Pannen aufgrund von Problemen mit der Batterie ergeben können, so ist auch die Ursache dieser Probleme zu hinterfragen. Hier wird häufig beim Start des Fahrzeugs festgestellt, dass die Batterieladung nicht mehr zum Starten ausreicht [Dude04]. An diesem Punkt werden die heutigen Probleme mit der Fahrzeugelektronik offensichtlich: es gibt eine Viel-

zahl von elektrischen Verbrauchern, die z.B. durch Fehlfunktionen in der Software die Busruhe nicht einhalten. An diesem Beispiel ist zu erkennen, dass auch die Steuergerätesoftware in Diagnoselösungen zu integrieren ist, weil sie nicht immer fehlerfrei funktioniert.

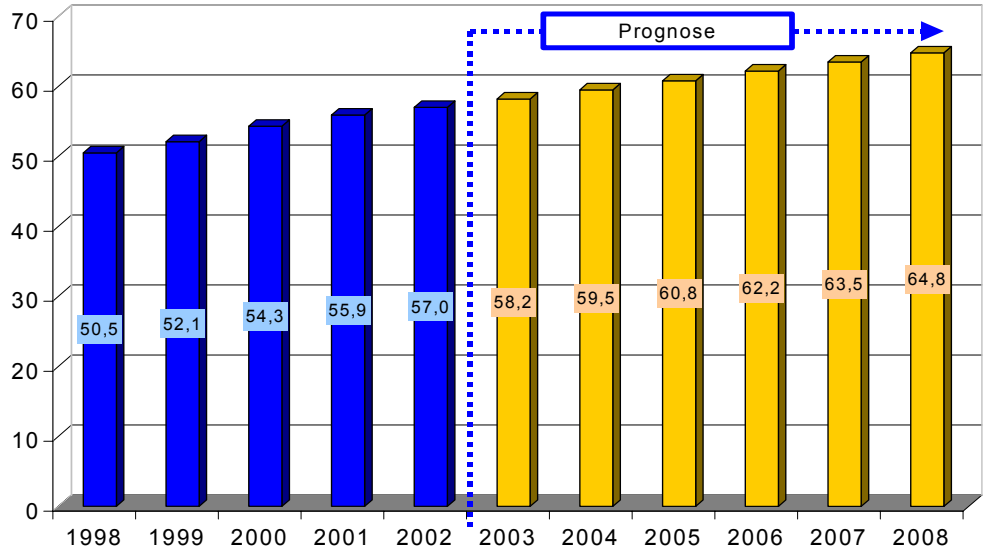


Abbildung 1-2 Anteil der Elektrik/Elektronik-Pannen steigt an [Dude04]

Die Diagnose von Software im laufenden Betrieb ist eng begrenzt auf die bloße Erkennung von Abweichungen vom Sollverhalten einzelner Funktionen. Der mögliche Einfluss von Fehlern auf andere Funktionen und andere Steuergeräte wird im Allgemeinen nicht systematisch betrachtet, d.h. die Fehlerfortpflanzungsmöglichkeiten sind nicht ausreichend bekannt. Es fehlt offensichtlich eine fahrzeugweite, systemübergreifende Diagnoselösung.

Bei den Fahrzeugherstellen werden daher zur Zeit verschiedene Ansätze verfolgt, um dieses Defizit zu beheben. Dabei wird vornehmlich der elektrische Kabelbaum des Fahrzeugs für weitergehende Diagnosen genutzt.

An dieser Stelle besteht die Möglichkeit, diese Herangehensweise zu erweitern, indem auch logische Abhängigkeiten in die Diagnoselösungen einbezogen werden, die sich aus der Software und der Kommunikation der Steuergeräte untereinander ergeben.

Ein wesentliches Ziel der Arbeit ist daher, durch die Reduktion auf die Frage „Wie hängen die Systemkomponenten voneinander ab?“ diese an sich völlig unterschiedlichen Teilsysteme miteinander zu verbinden und diese Informationen für die Diagnose nutzbar zu machen.

Durch die immer größere Zahl an Fahrzeugvarianten, die sich untereinander in Ausstattungsvarianten unterscheiden, stoßen konventionelle Werkstattdiagnosesysteme mit von Experten vorgegebenen fixen Fehlersuchstrategien an ihre Wartbarkeitsgrenzen.

Ein weiteres wesentliches Ziel der Arbeit ist es daher, einen Weg aufzuzeigen, wie sich ein Diagnosesystem für individuelle Fahrzeuge automatisch konfigurieren und parametrieren lässt, so dass eine fahrzeugspezifische Fehlersuchstrategie automatisch abgeleitet werden kann.

Um diese Automatisierbarkeit zu gewährleisten und gleichzeitig bereits früh im Entwicklungsprozess Diagnose betreiben zu können, sind Regeln für die Erstellung von Steuergerätesoftware zu erstellen, die ein derartiges Vorgehen unterstützen.

1.2. Gliederung

Die vorliegende Arbeit entstand im Rahmen des Forschungsprojektes STEP-X, das mit mehreren Instituten der TU Braunschweig und der Volkswagen AG bearbeitet wurde.

Zunächst wird im Abschnitt 2 die Basis der Arbeiten am Projekt STEP-X und auch die Basis für die vorliegende Arbeit, der modellbasierte Entwicklungsprozess näher erläutert, und darin beschrieben, in welchen Entwicklungsphasen welche Daten benötigt werden. Zusätzlich wird, da die Arbeit im Rahmen von STEP-X durchgeführt wurde, im Anschluss aufgezeigt, wie dieser Prozess von Seiten der Universität bearbeitet wurde, und welche Arbeitspakete sich daraus im Bereich der Fahrzeugdiagnose ergeben. Nähere Angaben zum Projekt finden sich in Abschnitt 2.9.

Die Arbeit beginnt in Abschnitt 3 mit einigen notwendigen Definitionen, um den Betrachtungshorizont für den Leser mit dem des Autors abzugleichen, da besonders im Bereich der Automobildiagnose häufig anwendungsbezogene und teilweise gegensätzliche Begriffsverständnisse existieren.

Anschließend soll im Abschnitt 4 ein Überblick über bisher im Einsatz befindliche Diagnoseverfahren und Werkstattsysteme gegeben werden, um darauf aufbauend eine Strategie zu erläutern, wie sich die aktuellen Probleme in der Automobildiagnose verringern lassen.

Im Anschluss wird der vorgeschlagene Diagnoseansatz im Abschnitt 5 ausführlich vorgestellt. Dabei wird zunächst auf die speziellen Vorzüge, Eigenschaften und Nachteile der bisher bekannten Diagnoseverfahren eingegangen, um daraus ein für die vorliegende Aufgabe optimales Verfahren zu entwickeln.

Im Verlauf von Abschnitt 6 wird dazu gezeigt, welche Informationen im Entwicklungsprozess anfallen, und synergetisch auch für die Zwecke der Diagnose genutzt werden können. Dabei wird auch näher dargestellt, wie der Entwicklungsprozess zu beeinflussen ist, um besser, d.h. automatisiert, Diagnoseinformationen aus den Entwicklungsmodellen ableiten zu können.

In Abschnitt 7 wird das betrachtete Demonstrations-Objekt des Komfortsystems eines Volkswagen Polos erläutert und anhand dessen werden die praktischen Ergebnisse des Diagnoseverfahrens dargestellt. Zum Vergleich werden in diesem Abschnitt bezüglich der Effizienz des Verfahrens Untersuchungen angestellt, welcher Entwicklungsmehraufwand sich für die Erstellung der Diagnose ergibt.

In Abschnitt 8 wird das bis dahin Beschriebene zusammenfassend dargestellt, und anhand eines Ausblicks gezeigt, welches Potential das Verfahren noch haben kann, wenn an verschiedenen Punkten weitere Verbesserungen oder Modifikationen vorgesehen werden, für die im Rahmen dieser Arbeit kein Raum mehr war.

2. Integration der Diagnose in die modellbasierte Entwicklung

Mit dem Begriff „Modellbasierte Entwicklung“ bezeichnet man einen Entwicklungsprozess, dem Modelle zugrunde liegen, mit denen sich die zu entwickelnde Funktionalität – unabhängig von real existierenden „Geräten“ – nachbilden lässt. Der besondere Vorteil liegt darin, dass man auf unterschiedlichen Abstraktionsebenen und aus verschiedenen Sichtweisen (z.B. Funktionsentwicklung, Test und Diagnose) entwickeln kann. Die Arbeiten im Bereich von STEP-X befassen sich mit der Bestimmung einer effizienten Methode, solche Entwicklungsmodelle zu erstellen, und in möglichst vielen Anwendungsgebieten der Automobilindustrie einzusetzen. Die für den Bereich Test und Diagnose erstellten Modelle sind in den meisten Fällen unabhängig von den Funktionsmodellen entstanden, was zu nicht abgedeckten Testfällen oder fehlerhaften Diagnosen führen kann.

Dies war die wesentliche Motivation für das Projekt STEP-X - mit Hilfe der modellbasierten Entwicklung die Modelle von Funktionsentwicklung, Test und Diagnose zusammen zu führen.

Die Funktionen werden im vorliegenden Fall von STEP-X grundsätzlich grafisch modelliert um ein übersichtliches Modell zu erhalten. Gleichzeitig bietet die grafische Repräsentation des Modells durch die enthaltene eindeutige Syntax der grafischen Elemente auch eine eindeutige Beschreibung des Modellverhaltens.

2.1. Definition Modell

Für die weiteren Ausführungen ist der Begriff des Modells einer Entwicklung von erheblicher Relevanz, da er im Rahmen von STEP-X und damit auch in dieser Arbeit eine große Bedeutung hat. Dabei bezeichnet ein Modell „eine Abstraktion eines Systems mit der Zielsetzung das Nachdenken über ein System zu vereinfachen, indem irrelevante Details ausgelassen werden“ [Brüg00]. Gegenüber herkömmlichen Entwicklungen auf Basis von textuell notierten Programmiersprachen, ist hier der Ansatz gewählt, aus einem definierten Satz von grafischen Komponenten mit genau vorhersagbarem Verhalten Systeme zu schaffen, die die Entwicklungsanforderungen exakt erfüllen.

Ein Modell beschreibt im Allgemeinen ein modular aufgebautes System aus Einheiten, die eindeutig definierte Eigenschaften besitzen und in ihrer Leistungsfähigkeit klar gegen andere Einheiten abgegrenzt sind. Diese Einheiten sind einzeln austauschbar, veränderbar und erweiterbar, ohne dass die restlichen Module des Systems dadurch beeinflusst werden. Zusätzlich können Komponenten des Systems auch aus weiteren Modellen zusammengesetzt sein.

Dadurch wird es möglich, in immer detaillierteren Abstraktionsebenen ein System genauer zu beschreiben. Im Bereich der objektorientierten Programmierung wird diese Eigenschaft konsequent ausgenutzt.

Seit die Entwicklungssysteme für Hard- und Software in den 1990ern immer leistungsfähiger wurden, ist auch die grafische Entwicklung unter Beibehaltung der objektorientierten Ansätze möglich. Vorteil der grafischen Notation gegenüber einer textuellen ist zum Einen die Übersichtlichkeit der Modelle. Zum Anderen ist die grafische Erstellung der Modelle frei von syntaktischen Fehlern, da die Entwicklungswerkzeuge lediglich zulässige Konstrukte anbieten. Vor allem aber ist die übersichtliche hierarchische Darstellung des Modells mit expliziten Signalpfaden der textuellen Notation weit überlegen. Durch die Verwendung von Modulbibliotheken mit aussagekräftigen Symbolen sind auch komplexe Systeme schnell zu erfassen. Damit sinkt u.a. auch die notwendige Einarbeitungszeit für neu hinzukommende Entwickler. Die Beschränkung bei der Modellerstellung auf nur einige wenige Grundelemente der Modellierungssprache mit klar definiertem Verhalten führt dazu, dass sich die entstehenden Systeme meist auch automatisiert in Bezug auf Determinismus, Determiniertheit, Vollständigkeit und Erreichbarkeit von Zuständen untersuchen lassen.

Ein Beispiel der grafischen Notation objektorientierter Entwicklungsmodelle ist die Unified Modeling Language (UML). Hierbei wurden in einem weit gefassten Industriegremium vierzehn verschiedene Beschreibungsmittel definiert[Seem00]. Mit Hilfe dieser Beschreibungsmittel sind die meisten diskret beschreibbaren Entwicklungen realisierbar. Für die Arbeiten im Projekt STEP-X wurde Artisan RTS von Artisan Software eingesetzt. Gegenstand der derzeitigen Definitionsmaßnahmen bei Weiterentwicklungen der UML ist die Integration von Echtzeitaspekten und kontinuierlichen Vorgängen in die Beschreibungsmittel. Die bisherige Beschränkung auf diskretes Verhalten und laufzeitunabhängige Beschreibung vereinfacht zwar mathematische Analysen zum Verhalten des Systems, allerdings sind damit regelungstechnische Systeme wie sie im Automobil an vielen Stellen vorherrschen oft nicht ausreichend beschreibbar.

Da an dieser Stelle eine allgemein akzeptierte grafische Notation von kontinuierlichen Funktionen für Entwicklungen im eingebetteten Bereich notwendig aber noch nicht verfügbar ist, sind in der Automobilindustrie regelungstechnische Blockschaltbilder im Einsatz. Weit verbreitet wird hier Matlab/Simulink von The Mathworks verwendet. Aus diesem Grund wurde dieses Softwarepaket für die kontinuierlichen Modellteile in STEP-X ebenfalls eingesetzt.

2.2. Anforderungen

Die Anforderungen im automobilen Umfeld Software zu entwickeln unterscheiden sich stark von allgemeiner Softwareentwicklung im PC-Bereich. Dies liegt zum Einen an den hohen produzierten Stückzahlen von Steuergeräten auf denen die Software zum Einsatz kommt, aber zum Anderen auch an den Sicherheitsanforderungen denen die entwickelte Software unterliegt. Immer muss sicher gestellt sein, dass die Qualität der Software ausreicht, um Gefahren auszuschließen und hohe Gewährleistungskosten im Falle von notwendigen Fehlerkorrekturen zu verhindern. Nur die Einhaltung eines nachvollziehbaren Produktentwicklungsprozesses kann die von den Automobilherstellern geforderte gleich bleibende Softwarequalität gewährleisten, die auch gesetzlichen Prüfungen standhält.

Prinzipiell handelt sich bei automobilen Systemen vor allem um verteilt eingesetzte und von unterschiedlichen Herstellern entwickelte eingebettete Systeme, die in einem Steuergerätenetzwerk betrieben werden. In einem Mittelklassewagen sind heutzutage zwischen 30 und 50 Steuergeräte verbaut [Scha05]. In diesem Zusammenhang wird die herstellerübergreifende Definition der Kommunikation der Steuergeräte untereinander und auch der Applikation mit dem Steuergerät immer weiter voran getrieben. Existierten bisher nur OEM-spezifische Lösungen, wird jetzt im AUTOSAR-Projekt herstellerübergreifend versucht, diese Schnittstellenproblematik aufzulösen [Scha05][Auto07]

Immer wichtiger wird in heutigen Entwicklungen die Frage nach einer Analysierbarkeit des Systems hinsichtlich Funktionalität und Sicherheit. Aus diesem Grund wird bei vielen Automobilherstellern ein modellbasierter Entwicklungsprozess installiert, der deutlich bessere Analysemöglichkeiten bietet, als die konventionelle Entwicklung einzelner Steuergeräte in einer Hochsprache [Mutz03][FORS05]. Nach [Mutz05] wird bereits die Erstellung der Spezifikation in den modellbasierten Entwicklungsprozess einbezogen, dessen Ablauf in den folgenden Abschnitten beschrieben ist.

Aufgrund des Kostenfaktors der hohen gefertigten Stückzahlen sind die auf den Steuergeräten zur Verfügung gestellten Ressourcen an Rechenleistung und Speicher nur gerade groß genug bemessen, um die eigentliche Aufgabe des Steuergerätes erfüllen zu können. Dennoch müssen die Systeme hoch zuverlässig sein, und Fehlfunktionen durch Überschreitung von zeitlichen oder anders gearteten Anforderungen ausgeschlossen werden. Neue Funktionalitäten, die mehr Rechenleistung, Buslast oder Speicher benötigen, sind daher nur schwer in bestehende Steuergeräte zu integrieren. Aufgrund dieser Erkenntnis spielt auch die zeitliche Analyse und

Verifikation der Kommunikationsstrukturen im Steuergerätenetzwerk eine wesentliche Rolle im Projekt STEP-X [Jers03]

Der modellbasierte Entwicklungsprozess beginnt bei allen Herstellern zunächst völlig ohne Modelle mit einer Spezifikation, in dem lediglich textuell die für ein spezielles Steuergerät umzusetzenden Funktionen und Kommunikationsschnittstellen beschrieben sind. Anhand des allgemein akzeptierten Entwicklungsprozesses nach dem V-Modell werden dann in der Anforderungsanalyse zunächst die dort aufgestellten Anforderungen identifiziert, um später im Grob- und Feinentwurf die Entwicklung auf dem Steuergerät zu implementieren. Analog zu diesen konstruktiven Entwicklungsphasen sind auf der rechten Seite des V-Modells jeweils auf entsprechender Ebene zugehörige Modul-, System- und Abnahmetests vorgesehen (s. Abbildung 2-1) [Drös00].

2.3. Erforderliche Systemsichten

Bereits zu Beginn der Produktentwicklung für ein automobiles Steuergerät ist eine Aufteilung der Betrachtungssichten auf das zu entwickelnde Produkt vorgesehen. So gibt es viele Details, die lediglich die Entwickler interessieren, und anderen Sichten, die nur ein Konstrukteur verwendet. Zur Einhaltung von Geheimhaltungsvereinbarungen ist z.B. auch eine angepasste Ansicht für Zulieferer vorstellbar. Eine weitere Sicht auf das System benötigt der Tester. Für ihn sind Implementierungsdetails irrelevant, denn für ihn zählen lediglich Anforderungen, deren Umsetzung überprüft werden muss.

Im Projekt STEP-X sind daher folgende Sichten vorgeschlagen worden, die sowohl Funktions-, wie auch Test- und Diagnoseumfänge beschreiben [Mutz03]:

- die *Benutzer- oder Produktebene*, die Zweck, Funktionalität und Randbedingungen eines zu entwickelnden Systems so beschreibt, wie es sich als Funktionsfamilie bei der Verwendung im Fahrzeug darstellt,
- die *Systemebene*, die Funktionalität, Schnittstellen und Implementierungsvorgaben auf der Ebene der technischen Realisierung des Teilsystems beschreibt,
- die *Subsystem- und Komponentenebene*, die je nach Komplexität des Produkts die Funktion und Interaktion von Teilsystemen in einem hierarchischen Entwurf beschreiben,
- die *Verifikationsebene*, die zu jeder genannten Ebene die Anforderungen zur Verifikation und zu Test und Diagnose enthält,

- die *Ebene der mitgeltenden Unterlagen*, die OEM-spezifische, gesetzliche oder dem technischen Stand zuzurechnende Vorgaben umfasst.

2.4. Abstraktionsebenen

Im V-Modell ist eine hierarchische Ordnung der Entwicklungsschritte vom Groben zum Feinen vorgesehen. Ein solches Vorgehen wird von allen grafischen, modellbasierten Entwicklungswerkzeugen unterstützt, indem hier die grafischen Modellkomponenten ebenfalls hierarchisch angeordnet werden. So enthält beispielsweise ein Block „Fensterhebersteuerung“ weitere Unterblöcke zur Priorisierung der Wünsche zwischen Fahrer und Beifahrer. In [Mutz05] ist für jede Abstraktionsebene des V-Modells dargestellt, welche UML-Sichten und Diagrammtypen genutzt werden sollten, um strukturiert, übersichtlich und für spätere Entwicklungen nachvollziehbar fehlerarme Modelle zu entwickeln.

2.5. Requirementsengineering

Wesentlicher Teil eines strukturierten Entwicklungsprozesses ist die Sicherstellung der Nachvollziehbarkeit der korrekten Umsetzung von der Anforderung bis hin zum umgesetzten Code auf dem Steuergerät mit Hilfe des Requirementsengineering. Werkzeuge, wie das in STEP-X eingesetzte DOORS, ermöglichen hier durch die Verwendung von Links direkt aus den Anforderungsdokumenten heraus Verweise auf die entsprechenden Umsetzungsstellen im Entwicklungsmodell zu verwenden. Durch die intensive Nutzung solcher Möglichkeiten wird auch die Änderung von Funktionsteilen leichter möglich, da die davon betroffenen Modellteile leicht identifiziert werden können [Mutz03].

2.6. Entwicklungsphasen

Im Folgenden werden die Entwicklungsphasen beschrieben, wie sie für das Projekt STEP-X aus dem V-Modell übernommen wurden (s. Abbildung 2-1)[Mutz03]. Sie bilden auch die Basis des Wissenerwerbs für die Diagnosekomponente.

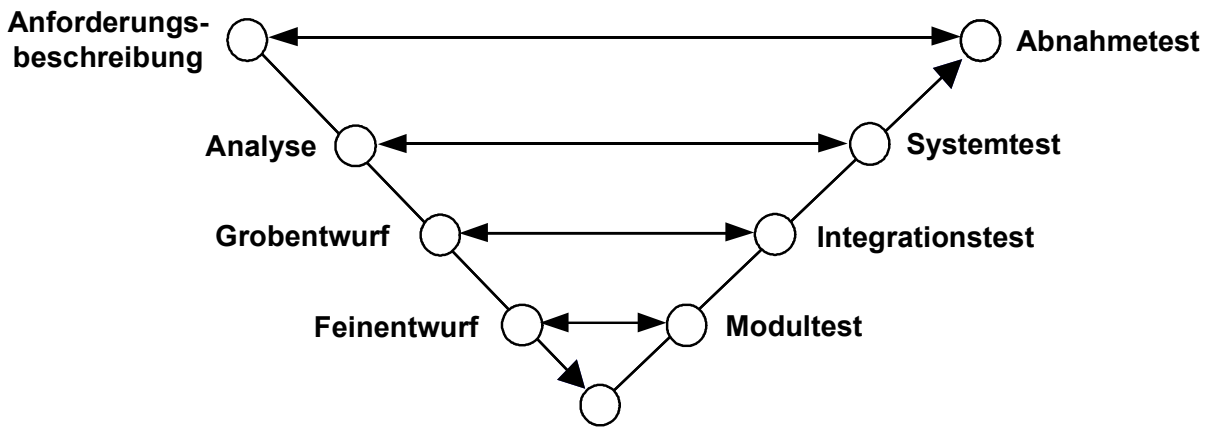


Abbildung 2-1 V-Modell zur Entwicklung von Hard-/Software

2.6.1. Anforderungsbeschreibung

Die Anforderungsbeschreibung bildet eine präzise, vollständige und zwischen allen am Produkt Beteiligten akzeptierte Grundlage für die strukturierte Entwicklung informationstechnischer Systeme. In der Fahrzeugindustrie werden daher schon seit längerem Spezifikationen, häufig in Form von Word-Dokumenten, verwendet, welche die Anforderungen an ein Subsystem im Fahrzeug ausführlich beschreiben. In diesen Spezifikationen wird die Funktionalität eines Subsystems, aber auch Zweck und Randbedingungen verschiedenster Art[1], umfassend dargestellt. Sie bilden die Basis für die Kommunikation zwischen den verschiedenen Entwicklergruppen beim OEM und beim Zulieferer. Da sich die Entwicklung dieser Subsysteme im Fahrzeug bisher an Bauteilen orientiert hat, beschreiben die Spezifikationen Funktionen in Hard- und Software sowie Eigenschaften eines Steuergerätes, wie es für eine Version eines oder mehrerer Fahrzeugtypen benötigt wird. In dem von uns strukturierten Entwicklungsprozess (STEP-X) wurden diese Spezifikationen unter Verwendung eines modernen Requirements Management und Engineering Tools in Digitale Spezifikationen überführt und in einigen Aspekten verbessert [Mutz05].

2.6.2. Analyse

In der Phase der Analyse und des funktionalen Grobentwurfs kommt als Beschreibungssprache die Unified Modeling Language (UML) zum Einsatz. Die UML ist eine grafische Modellierungssprache zum Spezifizieren, Entwerfen, Visualisieren und

[1] z.B. Abmaße, physikalische, chemische Verträglichkeit mit der Umgebung, gesetzliche und OEM-spezifische Regelungen, etc.

Dokumentieren von Softwaresystemen. Zahlreiche Diagrammarten ermöglichen unter Anderem eine Modellierung der Systemstruktur, des Systemverhaltens und der Kommunikation. Um die Menge der Beschreibungsmittel überschaubar zu halten, wird nur eine Teilmenge der UML-Diagramme genutzt und den verschiedenen Phasen zugeordnet.

In dieser Phase wird die Spezifikation auf Anforderungen an die Entwicklung analysiert. Dabei werden die zu realisierenden Funktionen identifiziert und als UML-Objekte erstellt. Die Interaktion des Systems mit der Umwelt wird aus der Spezifikation mit Hilfe des Anwendungsfalldiagramms notiert.

2.6.3. Funktionaler Grobentwurf

Im Funktionalen Grobentwurf wird zunächst mit Hilfe der UML-Objektdiagramme eine grobe Softwarearchitektur des zu erstellenden Systems aufgestellt. Auch grobe Abläufe können mit Hilfe von Sequenzdiagrammen dargestellt werden. Auf der Hardwareseite sind in einem Verteilungsdiagramm die Anzahl und Art der Steuergeräte und die Verbindung dieser durch einen Kommunikationsbus zu berücksichtigen. Die Funktionen aus dem funktionalen Grobentwurf werden in UML-Softwarekomponenten zusammengefasst und mit einer definierten Schnittstelle versehen.

Die so abstrahierten Softwarekomponenten werden bei der Partionierung anhand der technischen Randbedingungen aus der Hardware des Steuergerätenetzwerkes auf die logischen Steuergeräte verteilt. Für diese Verteilung der Funktionalität wird das Verteilungsdiagramm genutzt. Die einzelnen Steuergeräte werden in diesem Diagramm als Komponenten dargestellt.

Innerhalb der Komponenten werden die Funktionen als Objekte notiert. Gegebenenfalls muss aus Gründen der Übersichtlichkeit auf die Darstellung sämtlicher Objekte auf einem Knoten (d.h einem Steuergerät) in einem gemeinsamen Verteilungsdiagramm verzichtet werden. Die Informationen einzelner Knoten werden dann in separaten Diagrammen aufgezeigt.

2.6.4. Feinentwurf

Im Feinentwurf werden die Funktionen für den Serieneinsatz optimiert, indem die Modelle des funktionalen Grobentwurfs sukzessive detaillierter implementiert werden. Für die Modellierung im funktionalen Feinentwurf findet ein Notationswechsel von UML-Diagrammen zu Blockdiagrammen und einfachen Zustandsautomaten statt. Dies ist aus zwei Gründen notwendig. Einerseits ist die automatische und optimierte Codegenerierung für Mikrocontroller

mit UML-Werkzeugen zurzeit für die Serienentwicklung noch nicht geeignet, andererseits wird die Verteilung der Softwarekomponenten im funktionalen Grobentwurf in den Code nicht einbezogen. An dieser Stelle werden weitere Beschreibungsmittel notwendig, die insbesondere Aspekte der Kommunikation zwischen Teilfunktionen, der Partitionierung auf Hardware sowie des verwendeten Betriebssystems berücksichtigen.

2.7. Beschreibungsmittel

Bei der Umsetzung der Entwicklungsphasen des angewendeten V-Modells ist es notwendig, sich auf eine begrenzte Zahl der Beschreibungsmittel und Werkzeuge zu einigen, die für die Entwicklung genutzt werden.

Entwicklungsbegleitend zur Dokumentation der Anforderungen, der Umsetzung sowie der Testumfänge und Diagnoseinhalte wird hierfür ein Requirements Engineering und Management Werkzeug verwendet. Im Fall von STEP-X wurde hierfür DOORS von Telelogic ausgewählt.

Eine wesentliche Anforderung aus Sicht eines Automobilherstellers bei der Modellierung ist die Verwendung eines zukunftssicheren, herstellerunabhängigen und möglichst weit verbreiteten Standards. Für STEP-X wurde die UML als Modellierungssprache der ersten Entwicklungsphasen ausgewählt, weil sie diese Anforderungen am universellsten erfüllt. Viele Industrie-Standard-Werkzeuge bieten einen XML/XMI Import- und Export ihrer UML-Modelle an, so dass die Herstellerunabhängigkeit bei der Wahl des UML-Entwicklungswerkzeuges gewahrt bleibt. Durch den großen Umfang an Beschreibungsmitteln in dieser Sprache ist es allerdings erforderlich, sich auf einen bestimmten Satz von Beschreibungsmitteln und eine für alle an der Entwicklung Beteiligten verbindliche Entwicklungsmethode zu beschränken. Nur so können spätere Unklarheiten und Analyseprobleme wirksam verhindert werden. Durch die Reduktion der Anzahl der Beschreibungsmittel in der Analysephase kann ebenfalls gewährleistet werden, dass bereits Analysemodelle für frühe Modellerprobungen simulierbar sind.

Im Bereich der späten Entwicklungsphasen wird aufgrund der bisher nicht seriencodetauglichen Entwicklungswerkzeuge von der Modellierungssprache UML abgewichen. Ab dem Feinentwurf wird für kontinuierliche Steuerungsmodelle auf Matlab/Simulink durch seine Verbreitung als Industrie-Standard eingesetzt. Für diskrete Steuerungs-Anteile wird das Entwicklungswerkzeug Ascet-SD verwendet.

Für eine detaillierte Beschreibung der Entwicklungsmethodik und die zugehörigen Vorgaben für die Wahl der Beschreibungsmittel innerhalb des Projektes STEP-X wird an dieser Stelle auf [Mutz05] verwiesen.

2.8. Implementierung von Diagnosefunktionen

Die Implementierung der Diagnosefunktionen wird zu einem großen Teil bei der Nennfunktionsentwicklung anhand des V-Modells wie oben durchgeführt. Dabei werden Setzbedingungen für Fehlercodes anhand der Vorgaben aus der Spezifikation durch die Nennfunktionalität des Steuergerätes geprüft und modellbasiert implementiert. Ebenso ist die Reaktion auf außergewöhnliche Betriebszustände wie z.B. Fehler anwendungsspezifisch und damit Teil der Nennfunktion. Getrennt von der Anwendungsentwicklung werden jedoch die von außen zu erreichenden Steuergerätedaten und die Diagnosekommunikation über das Keywordprotokoll nach [ISO9141] verwaltet. Die für die Abfrage dieser Daten notwendigen Steuergerätefunktionen sind Bestandteil der so genannten Standardsoftware und müssen nicht für jedes einzelne Steuergerät neu implementiert werden. In den folgenden Abschnitten wird aufgezeigt, wie durch die konsequente Nutzung der Anforderungen aus der Spezifikation sowie der Entwicklungsmodelle der Anwendung die Diagnosefunktionen mit verhältnismäßig geringem Aufwand erstellt werden können und sich damit fest in den Entwicklungsprozess integrieren lassen.

2.9. Projekt STEP-X

Die Arbeit entstand im Rahmen des von der Volkswagen AG beauftragten Projektes STEP-X, das im Folgenden näher vorgestellt werden soll. Dazu wird zunächst die Intention von STEP-X beschrieben und im Anschluss die Aufteilung des Projektumfangs in einzelne Arbeitsgruppen und ihre Schnittstellen untereinander aufgezeigt, um die hier beschriebenen Arbeiten zur Diagnose in den Gesamtkontext einordnen zu können.

STEP-X ist ein Kooperationsprojekt der Volkswagen AG mit dem Zentrum für Verkehr (ZVB) der Technischen Universität Braunschweig und wird dort durch die Institute für Elektrische Messtechnik (emg, Fachbereich Elektrotechnik), für Software (ips, Fachbereich Informatik) sowie für Verkehrssicherheit und Automatisierungstechnik (iVA, FB Maschinenbau) bearbeitet, wodurch insgesamt drei verschiedene Fachbereiche der TU vertreten sind. Das Projekt war zunächst für eine Laufzeit von drei Jahren ausgelegt und hat Anfang 2001 begonnen. Durch den Erfolg der Untersuchungen wurde das Projekt unter Einbeziehung weiterer Gesichtspunkte und dem Institut für Datenverarbeitungsanlagen (IDA, Fachbereich Elektro-

technik) bis in das Jahr 2006 verlängert. Die Aufteilung der Arbeitsinhalte in fünf Arbeitsgruppen wurde anhand des bekannten V-Modells getroffen und ist in Abbildung 2-2 dargestellt. Im Folgenden sind die inhaltlichen Aufgabengebiete der abgebildeten Arbeitsgruppen näher erläutert.

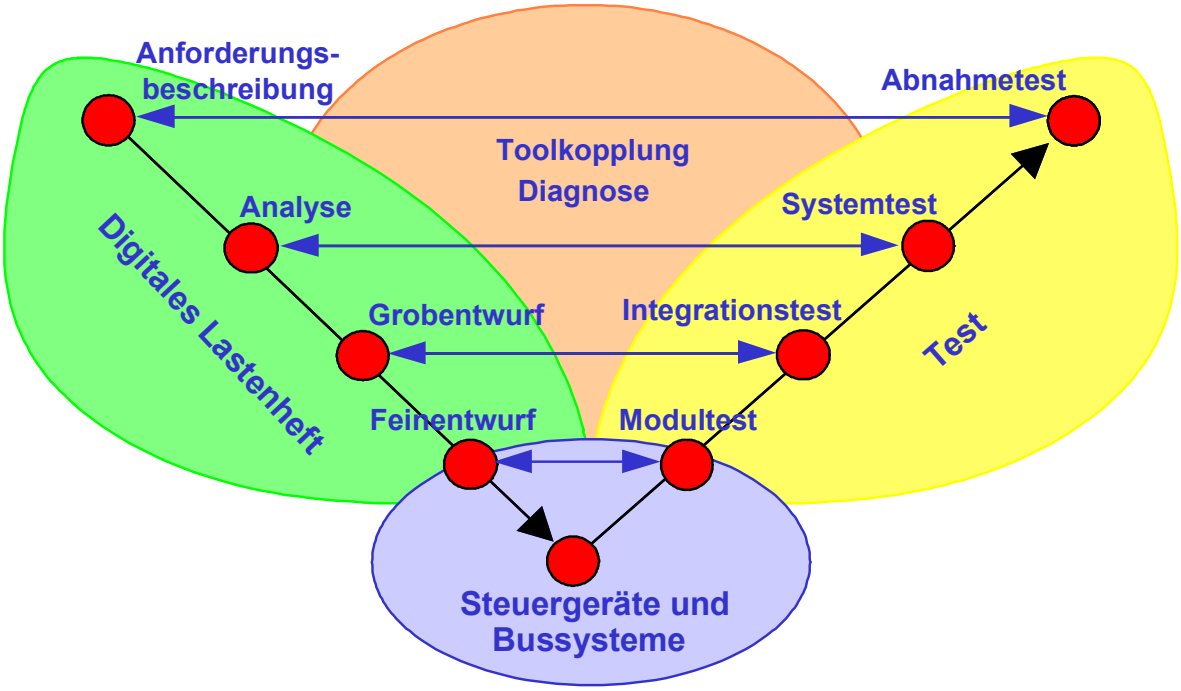


Abbildung 2-2: V-Modell und Arbeitsgruppenaufteilung in STEP-X

2.9.1. Digitale Spezifikation

Die Arbeiten zur digitalen Spezifikation beschäftigen sich vornehmlich mit der Definition einer gegenüber herkömmlichen Lastenheften in der Analysierbarkeit und Verwendbarkeit deutlich verbesserten Spezifikation. Ein wesentlicher Baustein des strukturierten Entwicklungsprozesses ist daher die gleichnamige digitale Spezifikation, die eine Vielzahl von Dokumenten der Anforderungsspezifikation und des Entwurfs auf System- und Komponentenebenen zusammenfasst. Zur digitalen Spezifikation gehören sowohl textuelle Anforderungsbeschreibungen, als auch modellbasierte Darstellungen der Struktur, des Verhaltens und der Kommunikation der zu entwickelnden Systemteile auf verschiedenen Detaillierungsebenen (siehe Abbildung 2-2). Neben der Darstellung der Funktionalität von Steuergeräten wurden auch nichtfunktionale Anforderungen wie gesetzliche und betriebliche Rahmenbedingungen berücksichtigt. Die digitale Spezifikation dokumentiert die Entwicklungsphasen von der Anforderungserfassung auf Systemebene über den Grobentwurf des Verhaltens, die logische Architektur sowie die Verteilung der Funktionen auf die jeweiligen Steuergeräte .

Die Verwaltung und Bearbeitung der Spezifikations-Dokumente erfolgt bei STEP-X mit dem Requirements-Management-Tool DOORS von Telelogic. Mit DOORS können Anforderungen und Teilmodelle verknüpft werden, um das Gesamtverständnis des betrachteten Systems zu verbessern und die Nachvollziehbarkeit der Entwicklung sicherzustellen. Als Fallstudie sind Funktionen aus dem Komfortbereich, wie Zentralverriegelung, elektrische Fensterheber und Spiegelverstellung gewählt worden. Die Anforderungen werden einheitlich und hierarchisch strukturiert und semantisch analysiert, um Vollständigkeit, Konsistenz und Eindeutigkeit der Beschreibung zu gewährleisten.

2.9.2. Toolkopplung

Wichtiger Aspekt zur langfristigen Wartbarkeit der erstellten Modelle ist die Verwendung von Standard-Entwicklungswerkzeugen anstelle von speziell für diesen Zweck entworfenen Hochschuleigenentwicklungen. Daher wurde ein Vergleich verschiedener kommerzieller CASE Tools durchgeführt, der zu dem Ergebnis geführt hat, dass keines der betrachteten Werkzeuge allen Phasen eines Entwicklungsprozesses für den Serieneinsatz im Automobil genügt. Da Methoden und Beschreibungsmittel im Vordergrund des vorgeschlagenen Entwicklungsprozesses stehen, ist Flexibilität bei der Werkzeugverwendung erwünscht.

Zur Lösung dieses Problems wird in STEP-X eine Kombination aus Co-Simulation und Code-Integration verfolgt. Die Co-Simulation erfolgt mit Hilfe des Werkzeuges EXITE der Firma EXTESSY und ermöglicht bezogen auf das V-Modell sowohl eine horizontale als auch eine vertikale Kopplung verschiedener CASE-Tools. Dadurch können neu entwickelte Steuerungsmodelle, die sich noch in der Analysephase befinden, mit existierenden Streckenmodellen gekoppelt werden, um zu einem frühen Zeitpunkt der Entwicklung simulativ die Funktionalität überprüfen zu können. Diese prototypischen Analysemodelle werden in späteren Phasen durch detailliertere Modelle ersetzt, die mit anderen Werkzeugen erstellt werden können.

Bei komplexen Modellen kann das Tool auch zur Performanzsteigerung der Simulation verwendet werden, indem das Modell auf mehrere Rechner verteilt wird. Zurzeit gibt es Schnittstellen zu den Werkzeugen Artisan RtS, Matlab/Simulink und Ascet SD, weitere entstehen bei Bedarf.

2.9.3. Steuergeräte und Bussysteme

Im Projekt werden durch die Fokussierung auf Komfortsystemfunktionen primär kontrollflussorientierte Systeme modelliert. In diesen werden typischerweise Daten asynchron ausgetauscht.

Um solche reaktiven Systeme über einen Bus kommunizieren zu lassen, sind ereignisgesteuerte Bussysteme gut geeignet, weshalb im Projekt der CAN-Bus eingesetzt wird.

Um unabhängig von verschiedenen Entwicklungswerkzeugen zu sein, und von diesen aus Quellcode problemlos in das Gesamtprojekt integrieren zu können, wird auf den Steuergeräten ein Betriebssystem nach dem OSEK/VDX-Standard verwendet. Es erlaubt die abstrakte Definition von Tasks, Nachrichten und Ressourcen aus verschiedenen Anwendungen heraus. Der Quellcode des Funktionsmodells wird dann zusammen mit all diesen Bausteinen übersetzt.

Auf dem CAN-Bus können alle Kommunikationsteilnehmer gleichberechtigt zu vorher nicht bekannten Zeiten Nachrichten verschicken. Wollen gleichzeitig Teilnehmer auf den Bus zugreifen, erhält derjenige Teilnehmer den Buszugriff, der die Nachricht mit der höheren Priorität verschicken möchte. Dies stellt ein Problem für die Validierung des Gesamtsystems dar, da die Rechtzeitigkeit der Übermittlung von Nachrichten nicht allgemein sichergestellt werden kann. Daher kann es bei einzelnen Steuergeräten bei hoher Busauslastung zu Nachrichtenstaus kommen, die zu Fehlfunktionen führen können. Aus diesem Grund wurde das Gesamtsystem langwierigen Tests unterworfen, deren Aussagen aber nur statistischer Natur sind, denn ein Test kann nicht wie eine Verifikation rechtzeitig eintreffende Nachrichten garantieren.

In dem Projekt wurden daher mit dem TTP/C auch ein synchrones Bussystem untersucht und in die Entwicklungsmethode integriert, u.a. mit dem Ziel die o.g. Überlasten bereits bei der Implementierung des Systems sicher ausschließen zu können. Diese Bussysteme arbeiten datenflussorientiert. Zeitgesteuerte Busse sollen in der Zukunft in sicherheitskritischen Systemen eingesetzt werden, in denen das Zeitverhalten des Systems garantiert sein muss. Durch die Aufteilung der Buszugriffe in Zeitscheiben, in denen jeweils ein Steuergerät zur Zeit exklusiven Buszugriff besitzt und auch durch die Definition der in diesem Zeitschlitz zu sendenden Daten bereits bei der Auslegung des Gesamtsystems, kann die Echtzeitfähigkeit des Systems garantiert werden.

2.9.4. Diagnose

Die Arbeiten in STEP-X zum Thema Diagnose werden in dieser Arbeit beschrieben. Unter Diagnose wird im Projekt sowohl die kontinuierliche Überwachung der entwickelten Steuergeräte während deren Betrieb im Fahrzeug als auch die Interpretation von Symptomen auf einem externen Testergerät verstanden. Dabei sollen die Diagnosefunktionen zum Einen

Fehlerzustände im laufenden Betrieb erkennen und betroffene Komponenten möglichst einwandfrei identifizieren. Zum Anderen wird für eine systemweite, d.h. steuergeräteübergreifende Diagnose ein abstraktes Modell der Kommunikationsbeziehungen und der elektrischen Verbindungen zwischen den Steuergeräten berücksichtigt. Identifizierte Fehler werden für eine spätere Verwendung z. B. in der Werkstatt geeignet auf den Steuergeräten abgelegt. Die Informations- und Modellierungsbedarf der Arbeiten zur Diagnose bezogen auf das V-Modell ist über alle Entwicklungsphasen hinweg gleichermaßen bedeutend, und wird daher in der Abbildung 2-2 als übergreifende Systemfunktionalität dargestellt.

Bei der Entwicklung von Steuergeräten werden im üblichen Entwicklungsprozess die Diagnosefunktionen erst nach der Implementierung der eigentlichen Nennfunktionen eines Systems implementiert. Dabei verlassen sich solche Diagnosefunktionen meist auf die Erfahrenheit von Entwicklern, kritische Situationen a priori zu definieren und dementsprechend zu behandeln.

Ausgehend vom modellbasierten Ansatz von STEP-X werden daher bereits in die Systembeschreibung in DOORS Vorschriften zur Erstellung der Diagnosefunktionen integriert. Durch die Verwendung von expliziten vorgeschriebenen Spezifikationsteilen, werden systematisch und teilautomatisiert abstrahierte Systeminformationen extrahiert und für die Generierung von Diagnosefunktionen verwendet. Dabei wird die Intention verfolgt, nach Möglichkeit wissensbasierte Systeme nicht zu verwenden, die vordefinierte Entscheidungsbäume enthalten da diese auf einem starren Fehlermodell basieren und einen großen Speicherbedarf haben.

Für die Berechnung der Diagnosemodelle auf den Steuergeräten gibt es qualitative und quantitative Ansätze zu denen verschiedene Werkzeuge auf Ihre Eignung hinsichtlich der Diagnosequalität untersucht wurden. Zusätzlich wurde eine weitere Methode untersucht und in dieser Arbeit beschrieben, die sich mit Abhängigkeiten befasst, die sich aus Strukturmodellen ergeben. Dieses Vorgehen hat gegenüber modellbasierten Ansätzen den Vorteil des geringeren Aufwands bei der Erstellung der diagnostischen Wissensbasis.

2.9.5. Test

Zusätzlich zur Entwicklungsmethodik als konstruktivem Teil der Qualitätssicherung stellt der Test den wichtigsten Teil der analytischen Qualitätssicherung in STEP-X dar. Die Problemstellung für das Testen ist typisch für eingebettete Systeme der Verkehrstechnik und ergibt sich aus den Systemrandbedingungen einer engen Kopplung zwischen Umgebungshardware,

Elektronikhardware und Software, einem hohen Grad an Vernetzung sowie einem kontinuierlichem als auch diskretem Charakter, wodurch die Anzahl der möglichen Testfälle beliebig groß wird.

In STEP-X wird nach dem V-Modell zwischen Modul-, Integrations- und Abnahmetest unterschieden. Ziel war es, eine eng an den Entwicklungsprozess angelehnte praktikable Testmethodik zu definieren, bei der zunächst Abnahmetests der in der Spezifikation geforderten Funktionalitäten durchgeführt werden. Je nach Kritikalität und vorhandenen Ressourcen werden zusätzliche Testfälle ermittelt und ausgeführt. Der funktionale Abnahmetest hat verschiedene Abstraktionsebenen des Systems als Ziel, begonnen beim ersten ausführbaren Analysemodell über das Designmodell bis zur realen Implementierung, wobei jeweils die gleichen Testsuiten verwendet werden können.

2.10. Ansatz der Arbeit

Die realisierte Diagnoselösung hat als vorrangiges Ziel, sich nahtlos in den bestehenden modellbasierten Systementwurf für eingebettete Systeme zu integrieren, und dadurch anfallende Entwicklungsinformationen effizient für die Diagnose zu nutzen. Dafür musste ein Diagnosealgorhythmus gefunden werden, der sich für die vorgeschlagene enge Integration in den Entwicklungsprozess eignet, und dabei gleichzeitig die Variantenvielfalt heutiger Fahrzeuge beherrscht.

3. Terminologie

Im Folgenden wird die Basis der durchgeführten Arbeiten beschrieben. Zunächst soll der dem Verfahren zugrunde liegende Begriff der modellbasierten Entwicklung näher erläutert werden, bevor die für die Diagnose im Automobilbereich notwendigen Definitionen der verwendeten Begriffe und Methoden erfolgen. Im Anschluss daran werden die bisher verfügbaren Ansätze zur Diagnose erläutert sowie die für den Werkstattbereich verfügbaren Diagnosesysteme beschrieben und die Grenzen ihrer Anwendbarkeit aufgezeigt.

3.1. Modellbasierte Entwicklung als Grundlage der Diagnose

Modellbasierte Entwicklung bezeichnet die generelle Entwicklung von Funktionalität mittels zugrunde liegender Modelle. Diese sind im vorliegenden Fall von STEP-X grundsätzlich grafischer Art. In der grafischen Repräsentation des Modells ist über die vordefinierte Semantik der grafischen Elemente das Verhalten des Modells eindeutig definiert (falls eine eindeutige Semantik vorhanden ist).

Vorteil der modellbasierten Entwicklung gegenüber textueller Entwicklung ist einerseits die von syntaktischen Fehlern freie Erstellung von Modellen, da die grafischen Entwicklungstools lediglich zulässige Konstrukte anbieten. Andererseits ist die übersichtliche hierarchische Darstellung des Modells mit expliziten Signalpfaden der textuellen Notationen überlegen. Damit sinkt auch die notwendige Einarbeitungszeit neu hinzukommender Entwickler.

3.2. Diagnose – Ziele und Nomenklatur

Das Thema Diagnose im Fahrzeug hat die in den USA in Kalifornien seit 1988 vorgeschriebene so genannte OBD-I-Schnittstelle (OBD: On-Board Diagnosis) stark voran getrieben [ISO9141]. Vorhergehende Ansätze waren lediglich von den unterschiedlichen Herstellern realisierte Insellösungen. OBD hingegen ist ein herstellerübergreifender Standard. Dabei handelt es sich um ein System, das zum Einen den Fahrzeugführer per Kontrolllampe im Cockpit auf den nicht optimalen Verbrennungsvorgang und damit eine erhöhte Umweltbelastung hinweist. Damit soll der Fahrer zum Besuch einer Werkstatt bewegt werden, die den Mangel beheben kann. In folgenden Vorschriften wurden zum Anderen ein Stecker, eine elektrische Schnittstelle und ein Protokoll vorgeschrieben über die Zugang zu Motordaten gewährt werden muss. Damit wird die Fehlersuche in den Werkstätten universell vereinfacht und ohne Einsatz von zusätzlichen Messgeräten werden die nur den Steuergeräten zugänglichen Messdaten nach außen transportiert. Bei den zur Verfügung gestellten Daten handelt es sich bei-

spielsweise um Lambdasondenmesswerte, Drosselklappenstellung und Temperaturen [ISO9141]. Durch die Definition eines Standards für die Abfrage der Diagnose-Daten mit Hilfe eines so genannten Scan-Tools wurde auch freien Werkstätten der Weg zu den Steuergeräten der verschiedenen Hersteller geebnet. Auch in Europa wurde 1996 dieser Standard für alle Neufahrzeuge in Anlehnung an das amerikanische Beispiel eingeführt. In Europa heißt der entsprechende Standard EOBD für European-On-Board-Diagnosis.

3.2.1. Begriffsbildung

Im Bereich der Fahrzeugelektronik und im allgemeinen Sprachverständnis gibt es sehr unterschiedliche Auffassungen, was mit dem Schlagwort Diagnose bezeichnet wird. So gibt es Diagnosen nicht nur in der Medizin, sondern auch in der Anlagentechnik und vielen anderen Fachgebieten. Für diese Arbeit am wichtigsten ist die Begriffsdefinition aus der Automobiltechnik. Gerade hier aber gibt es sehr unterschiedliche Auffassungen, was Diagnose bedeuten kann. Aus diesem Grund soll in diesem Abschnitt zunächst eine Aufstellung der gebräuchlichen Themen erfolgen, die unter den Diagnosebegriff fallen. Abgrenzend wird dann definiert, wie der Begriff der Diagnose im Rahmen dieser Arbeit verstanden werden soll.

Für den Fahrzeug-Hersteller beschreibt es einerseits die „Diagnosekommunikation“, d.h. Flashen, Parametrieren und Auslesen des Fehlerspeichers und andererseits die Definition möglicher Fehlerspeichereinträge und den mit dem Steuergerät austauschbaren Datenumfang. In dieser Beziehung bedeutet Diagnose daher sowohl das Ergebnis einer Fehlerfindung, als auch den allgemeinen Datenverkehr mit den Steuergeräten in der Werkstatt.

Für den Steuergeräteentwickler auf Seiten des Zulieferers bezeichnet „Diagnose“ vor allem die „Steuergeräteigendiagnose“, d.h. das einzelne Steuergerät registriert Fehler in seiner Umwelt bzw. an seiner Hardware und generiert entsprechende Fehlerspeichereinträge. In diesem Kontext wird also ein Befund für eine gegebene Situation erstellt und gespeichert. Die Steuergeräteigendiagnose ist damit ein Teil der Diagnose, der im Fahrzeug stattfindet, und damit ein Teil der so genannten Onboard-Diagnose.

Für den Kundendienst und die Werkstätten sind diese Aspekte weniger wichtig. Dort muss basierend auf Diagnosekommunikation, Sensor- und Stellgliedtests sowie Fehlerspeichereinträgen die eigentliche Fehlerursache gefunden werden, damit ein Mangel behoben werden kann. Daher werden hier lediglich die Diagnosebefunde der Steuergeräte verwendet und analysiert.

Um die Vielfalt der Begriffsdeutungen einzuschränken, definiert daher der VDI den Begriff der Diagnose beispielsweise für speicherprogrammierbare Steuerungen. Dabei bezeichnet Diagnose das „Erkennen fehlerhafter Steuerungszustände, Abläufe und ihre Ursachen“ [VDI85].

Angelehnt an diese Auffassung wird abgrenzend für das automobile Umfeld in dieser Arbeit definiert:

Definition Diagnose

Die Diagnose ist der Prozess des Erkennens und Deutens fehlerhafter Zustände und Abläufe im Kraftfahrzeug.

Damit ist Diagnose der Prozess, durch den der Status des Fahrzeugs bestimmt wird. Im Falle von vorhandenen Fehlern werden diejenigen Komponenten identifiziert, die das Fehlverhalten verursachen. Das für den Kundendienst nutzbare Diagnoseergebnis besteht aus den fehlerverursachenden Komponenten.

Damit sind alle wesentlichen Sichten auf das System in die Definition integriert: nur mit Hilfe standardisierter Diagnosekommunikation kann ein allgemein gehaltenes Diagnosesystem auf Daten des Fahrzeugs zurückgreifen. Dabei bildet die Steuergeräteigendiagnose die Basis des Wissenerwerbs für die Diagnose.

Für den Kundendienst hergestellte Diagnosesysteme funktionieren meist anhand der so genannten geführten Fehlersuche. Dabei muss ein Werkstattmitarbeiter eine vom Diagnosesystem vorgegebene Schrittfolge von Handlungsanweisungen befolgen. Die Schrittfolge wird anhand der Informationen angepasst, die aus den einzelnen Prüfschritten gewonnen werden, um die Fehlersuche möglichst kurz zu gestalten. Die geführte Fehlersuche wird im Allgemeinen in der Werkstatt auf speziell dafür eingerichteten Computern durchgeführt, die über eine Kommunikationsschnittstelle mit dem Automobil verfügen.

Definition Fehler

Nichterfüllung einer festgelegten Forderung [ISO8402].

Ein Fehler liegt daher vor, wenn eine Funktionseinheit die anhand von Referenzwerten spezifizierte Anforderung in ihrem Verhalten nicht erfüllt.

Definition Fehlerkandidat

Eine mögliche Erklärung für das Vorliegen eines Fehlers.

Damit ist ein wesentliches Ziel der Diagnose die Nennung der Menge der Fehlerkandidaten, um Fehler beheben zu können.

Anzustreben ist, dass die Menge der Fehlerkandidaten möglichst deckungsgleich mit den tatsächlich im System vorhandenen Fehlerursachen ist. Werden zu viele Fehlerkandidaten ermittelt, sind die Kosten durch detailliertere Prüfungen und unnötige Bauteilwechsel in der Werkstatt zu hoch. Bei zu kleiner Fehlerkandidatenliste werden nicht alle Fehlerursachen ermittelt, und demzufolge muss der Werkstattmitarbeiter ohne Anleitung durch das Diagnosesystem auf eigene Faust versuchen, den Fehler zu beheben, was mit höheren Standzeiten in der Werkstatt und ebenfalls dem möglichen Austausch von eigentlich korrekt funktionierenden Komponenten verbunden sein kann.

Aufbauend auf der Liste der Fehlerkandidaten muss ein Diagnosesystem die jeweiligen betroffenen kleinsten Einheiten bestimmen, die in der Werkstatt repariert bzw. getauscht werden können, um das System wieder in den Normalzustand zu versetzen. Beispielsweise wird ein als defekt erkannter Treibertransistor innerhalb eines Steuergerätes nicht ausgelötet und getauscht, stattdessen wird der Austausch des kompletten Steuergerätes erforderlich.

3.2.2. Anforderungen an das Diagnosesystem

Die Anforderungen an ein Diagnosesystem sind wie an alle technischen Systeme teilweise gegensätzlich. Es gilt daher immer, den für die jeweilige Anwendung am Besten passenden Kompromiss zu finden. Generell lassen sich aber folgende Kriterien beschreiben, anhand derer sich unterschiedliche Diagnoselösungen messen lassen können [Lemm94].

3.2.2.1. Anforderungen an die Leistungsfähigkeit

- Verarbeitungszeit der Diagnosefunktionen

  Die Verarbeitung der Diagnosefunktionen soll möglichst schnell erfolgen, um zum Einen Ressourcen an Bord des Fahrzeuges zu sparen, und zum Anderen in der Werkstatt Personalkosten und Standzeiten zu minimieren.

- Nachvollziehbarkeit der Ergebnisse und Zwischenschritte

  Es sollte erkennbar sein, aus welchem Grund einzelne Schritte zur Problembehebung beitragen und was die vermutete Ursache der Fehlfunktion ist.

- Abdeckungsgrad des diagnostizierten Systems

  Das System sollte möglichst gänzlich erfasst und diagnostizierbar sein, so dass alle auftretenden Fehler erklärbar sind.

- Rand des diagnostizierten Systems

  Nach Möglichkeit sollte das System sowohl interne Fehler, als auch Fehler in seiner Umwelt korrekt erkennen können und entsprechende Fehlerbehebungsvorschläge generieren können.

- Flexibilität des Systems

  Änderungen am System und Korrekturen sollten möglichst leicht in das Diagnosesystem einzupflegen sein.

- Exaktheit der Ergebnisse

  Das ermittelte Diagnoseergebnis sollte möglichst nur den tatsächlichen Fehler erklären, und Alternativfehler ausschließen.

3.2.2.2. Generelle Anforderungen an ein Diagnosesystem

Die Aufgabe eines automatisierten Diagnosesystems ist die Bestimmung von plausiblen Erklärungen für das beobachtete Fehlverhalten eines Systems. In vielen Fällen ist es notwendig, die tatsächlichen Fehler in den Komponenten derart zu identifizieren, dass die Beobachtungen als Konsequenzen dieser Fehler angesehen werden können (Fehleridentifikation). Trotzdem ist es in einigen Fällen ausreichend, die fehlerhaften Komponenten lediglich zu isolieren (Fehlerisolation). Dabei wird ein Satz von Systemkomponenten bestimmt, die unter der Annahme, dass keine der bestimmten Komponenten korrekt funktioniert, die Beobachtungen konsistent erklären.

Für eine Fehleridentifikation muss bereits bei Erstellung der Fehlerfindungsstrategie beschrieben werden, wie die Komponenten sich im Fehlerfall verhalten. Im Gegensatz dazu erfordert eine Fehlerisolation lediglich die Beschreibung des korrekten Verhaltens einer Komponente. Von der wissensbasierten Diagnose wird die Fehlererkennung, also die Tatsache, dass überhaupt ein Fehler vorliegt, nicht zur eigentlichen Diagnose gerechnet, sondern als Teil der gegebenen Systemfunktionen betrachtet.

Unabhängig von der Aufgabe, Fehlerisolation oder Fehleridentifikation zu betreiben, kann die Performanz eines Diagnose-Systems anhand folgender Kriterien bestimmt werden:

- Vollständigkeit

  das Diagnosesystem soll eine vollständige Liste von möglichen Diagnosen erstellen. Speziell sollte es möglich sein, Einzel- wie auch unabhängige Mehrfachfehler diagnostizieren zu können. Bei der Suche nach plausiblen Diagnosen sollte die Vollständigkeit der Effizienz vorgezogen werden.

- Robustheit

  Wichtig für jedes Diagnosesystem ist die vorausschauende Implementierung des Diagnoseverfahrens. Änderungen während der Entwicklung oder während der Serienproduktion müssen leicht einpflegbar sein.

- Effizienz

  Die Diagnosen müssen schnell genug erfolgen. Für einen Einsatz an Bord eines Fahrzeugs sind der Rechenlast und dem Speicherbedarf für die Diagnosesysteme enge Grenzen gesteckt.

- Skalierbarkeit

  Das System darf nicht nur für Demonstrationsanwendungen geeignet sein, sondern muss auch der Komplexität in tatsächlichen automobilen Anwendungen gerecht werden.

- Generelle Verwendbarkeit

  das System sollte leicht auf unterschiedliche Klassen von Systemen anwendbar sein.

- Entwicklungsaufwand für die Diagnose

  Der zusätzliche Entwicklungsaufwand für das Diagnosesystem sollte minimal sein. Nach Möglichkeit sollten Daten aus dem Entwicklungsprozess der Nennfunktion auch für die Diagnose genutzt werden.

- Automatisierbarkeit

  Die individuelle Parametrierung eines Diagnosesystems auf ein spezifisches Fahrzeug sollte möglichst automatisiert erfolgen können. Die a priori Parametrisierung aller möglichen Fahrzeugausstattungskombinationen ist aufgrund seiner unüberschaubaren Anzahl nicht realisierbar.

- Universelle Einsetzbarkeit

  Das Diagnosesystem sollte anwendungs- und anwenderunabhängig realisiert sein.

- Hardwareaufwand

  Ein Diagnosesystem sollte auf Basis vorhandener Sensorwerte zu möglichst präzisen Ergebnissen kommen. Zusätzlicher Verbau von Sensorik oder Steuergeräten allein für Diagnosezwecke ist aus wirtschaftlichen Gründen fast immer ausgeschlossen.

3.2.3. Randbedingungen für den Anwendungsfall Kraftfahrzeug

Neben den allgemeinen Anforderungen an eine Diagnoselösung, präzise Fehler und fehlerhafte Komponenten zu ermitteln, ergeben sich durch den Einsatz im Kraftfahrzeug weitere Einflüsse, die in den Betrachtungen zu berücksichtigen sind.

Laut der freien Internetenzyklopädie Wikipedia ist der Begriff Mechatronik (Mechanik-Elektronik) ein Kunstwort. Er wurde ab 1969 von der japanischen Firma Yaskawa Electric Cooperation geprägt und findet seinen Ursprung in der Feinmechanik. Später kam die Informatik als neue Kerndisziplin hinzu [Wiki05]. In Abbildung 3-1 ist diese Kopplung der unterschiedlichen Domänen gut zu sehen, und die Verbindung zum Automobil wird selbst erklärend.

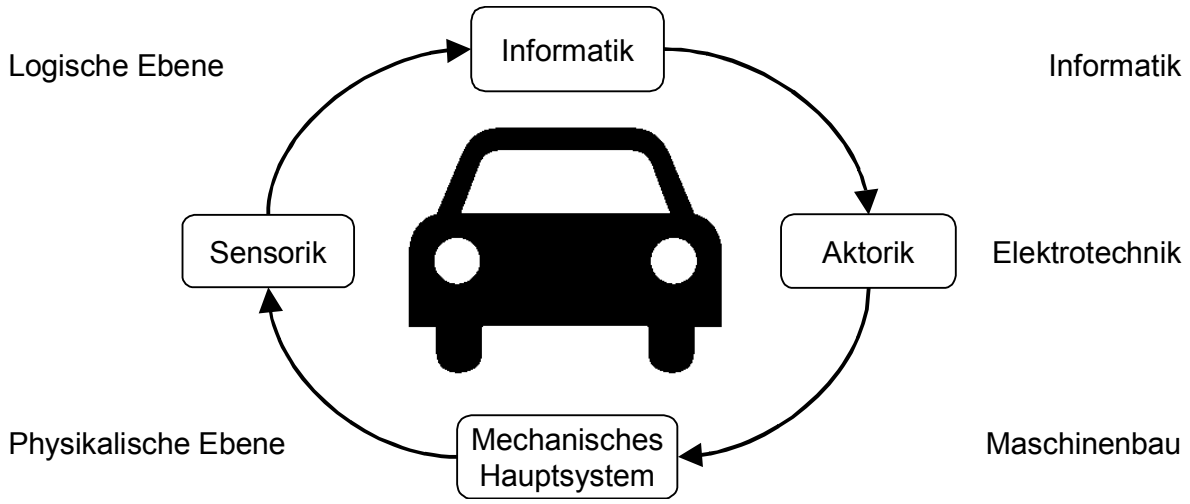


Abbildung 3-1 Das Kraftfahrzeug als mechatronisches Gesamtsystem

Weiteres wichtiges Element, das in die Betrachtungen eingeht, ist die große Stückzahl von Steuergeräten, die mit hohem Kostendruck in den Fahrzeugen verbaut werden. Für Käufer von Fahrzeugen sind die sichtbaren Faktoren eines Kraftfahrzeugs vor allem der Markenstatus, der Fahrzeugtyp und die Ausstattung. Die funktionierende Diagnose eines Kraftfahrzeugs ist daher nicht Hauptinteresse eines Automobilherstellers, sondern lediglich ein Mittel, den Werkstattbesuch eines Kunden möglichst kurz und damit kostengünstig zu gestalten. Dement-

sprechend ist die Diagnose bisher nur an den konstruktiven Entwicklungsprozess angehängt, und fristet ein Nischendasein.

Dennoch gelangt eine Fehlfunktion der Anwendungsfunktionen schnell in das Bewusstsein der Fahrzeugbenutzer. So gibt es von vielen Anwendern Geschichten von Werkstattbesuchen, in denen lediglich die Fehlerspeicher gelöscht wurden, weil kein eigentlicher Fehler gefunden wurde. In anderen Fällen lernen die Fahrer bestimmte Fehleranzeigen zu ignorieren, weil ihnen bei den dann bereits bekannten Fehlerbildern in der Werkstatt sowieso nicht geholfen werden kann.

Eines der wesentlichen Probleme bei der Erstellung von Diagnosesystemen ist daher die Wissensbeschaffung. Schwierig ist hierbei vor allem die notwendige Verwendung und Mischung heuristischen Wissens, empirischer Assoziationen sowie fachspezifischen Wissens [Rich93]. Diese umfassende Betrachtung ist notwendig, da die zu diagnostizierenden Systeme im Kraftfahrzeug nur teilweise durch physikalische Sensormessgrößen fundiert sind. Weitere wesentliche Eingangsinformationen sind Erfahrungen von Werkstattmitarbeitern und „weiche“ Eingangsgrößen wie „etwas klappert“ oder „Schwarzrauch aus der Abgasanlage wird beobachtet“. Weitere für den Fahrzeughersteller notwendige Eingangsgröße zur Erstellung des Diagnosesystems ist die Betrachtung von Arbeitsaufwand zum Tausch bzw. zur Prüfung eines Bauteils.

3.2.3.1. Häufige auftretende Fehler und ihre Ursachen

Die häufig auftretenden Fehler sind zunächst grundsätzlich in zwei unterschiedliche Kategorien einzuordnen. Denkbare Fehler, die bereits während der Montage auftreten können, und bei der Endabnahme vor der Übergabe des Fahrzeugs an den Kunden erkannt werden müssen. Andere Fehler entstehen erst im Laufe des Fahrzeuglebens aufgrund von Umwelteinflüssen, Vibrationen etc. im Fahrbetrieb.

Während der Montage mögliche Fehler können unkorrekt aufgesteckte Steckverbindungen sowie beschädigte Kabel im Kabelbaum sein. Tabelle 3-1 zeigt mögliche Fehlerquellen zusammen mit ihren elektrischen Auswirkungen.

Ursache	Wirkung	Elektrische Auswirkung
Unsachgemäße Montage des Kabelbaums	Isolationsschäden	Schluss zur Fahrzeugmasse
Fehlerhafte Montage von Steckern	Verbogene Pins	Schlüsse zwischen benachbarten Pins
Vertauschung von Signalen im Kabelbaum	Bauteildefekte, falsche Signalinterpretation	Verschiedene Auswirkungen
Bauteilfehler	Verschiedene Auswirkungen	Verschiedene Auswirkungen

Tabelle 3-1 Typische Fehler an Elektrik/Elektronik im Automobil bei der Montage [Spit01]

In den mechatronischen Fahrzeugsystemen auftretende Fehler sind vor allem durch die rauen Umgebungsbedingungen im Automobil gekennzeichnet. Als häufigste Elektrik- und Elektronik-Fehler werden mit ca. 40% Leitungs- und Steckverbindungsdefekte angegeben. Die angeschlossene Aktorik ist in 26% und die Sensorik in 18% der Fälle fehlerhaft. Die Steuergeräte selbst sind in nur 16% der auftretenden Fehlfunktionen als defekt einzustufen [Ring99][Beil96]. Diese Zahlen aus dem Jahr 1996 decken sich auch noch mit aktuellen Erfahrungen in der Werkstatt. In Gesprächen mit Kundendienstmitarbeitern einer Volkswagen-Vertragswerkstatt wurden vornehmlich lose Stecker als Fehlerursache ausgemacht. Seltener seien einzelne korrodierte Pins in Steckverbindungen, und noch geringer die Anzahl an Leitungsbrüchen und –kurzschlüssen. Häufig werden Defekte ohne eindeutige Fehlerursache durch den Tausch von Steuergeräten behoben, und verschwinden danach, ohne dass eine konkrete Fehlerursache gefunden wurde.

Im Automobil sind Vibrationen allgegenwärtig und die Ursache vieler Ausfälle. So werden z.B. nicht gänzlich verriegelte Steckkontakte mit hoher Wahrscheinlichkeit irgendwann im Lauf der Zeit Wackelkontakte produzieren oder gänzlich die Verbindung lösen. Ebenso können durch stetige Vibration oder unsachgemäßen Einbau elektrische Leitungen beschädigt werden, so dass Leitungskurzschlüsse zur Fahrzeugmasse auftreten können. Durch ähnliche Einflüsse entstehen Leitungsbrüche.

Ebenso sind im Automobil Verschmutzungen durch aufgewirbelten Staub und Abgase allgegenwärtig. Aufgrund des vermehrten Einsatzes von verriegelnden und spritzwassergeschützten Steckverbindungen sowie verbesserten Kontaktfetten ist die Fehlerzahl durch derartige Ursachen geringer geworden.

Die meisten aufgetretenen Fehler bezogen sich auf mechanische Fehler, die aufgrund von Verschleiß aufgetreten sind. In diesen Fällen ist die Fehlerursache meist offensichtlich: z.B.

können elektrische Antriebe aufgrund von übermäßigem Schlupf ihre Funktion nicht ausreichend ausführen.

Auch das Alter eines Automobils kann die korrekte Funktion eines Fahrzeugs beeinflussen. Zu Beginn der Fahrzeuglebens verwendete Modelle für Fahrzeugkomponenten beschreiben nicht mehr den veränderten, weil gealterten tatsächlichen Komponentenzustand, d.h. Kennfelder sind nicht mehr genau passend, oder es können sich Widerstände und Bauteil-Toleranzen verändern.

Zusammenfassend sind in Tabelle 3-2 die Hauptursachen für Defekte am elektrischen System während des Betriebs eines Fahrzeugs mit ihrer jeweiligen Auswirkung genannt.

Ursache	**Wirkung**	**Elektrische Auswirkung**
Spritzwasser, Staub	Korrosion	Wackelkontakte, Änderung des Übergangswiderstands
Alterung	Isolationsschäden	Leitungsschlüsse
austretende Flüssigkeiten: Öle, Treibstoffe, Bremsflüssigkeit, Batteriesäure	Isolationsschäden	Leitungsschlüsse
Vibration	Öffnung von Kontakten	Wackelkontakte, Lösen von Steckverbindungen, Ausgebrannte Kontakte
	Schäden an der Isolation	Leitungsschlüsse
Mechanische Defekte (Vibration, Unfall, etc.)	Schäden an Isolation und Leitung	Leitungsbruch oder Leitungsschlüsse
Alterung	Bruch kalter Lötpunkte, Kontaktkorrosion	Steuergerätedefekt
Alterung	Leitungsmantelschäden	Leitungsschlüsse
Temperaturschwankungen	Beschleunigte Korrosion, Mechanische Belastung	Verschiedene Auswirkungen
Ionisierende Strahlung	Fehler in RAM	Software Runaway Faults

Tabelle 3-2 - Typische Fehler an Elektrik/Elektronik im Kfz im Betrieb [Hart01][Spit01][Maso05]

3.2.3.2. Variantenbildung und kleinste tauschbare Einheit

Heutzutage hat der Kunde eine sehr große Auswahl bei der Ausstattung seines Fahrzeugs, woraus sich eine Vielzahl von Varianten ergibt, die ein einzelner Fahrzeugtyp ausbilden kann. Für alle diese Varianten muss eine Diagnose durchführbar und zielführend sein.

Auch der Einsatz unterschiedlicher Soft- und Hardwareversionen ist für die Diagnose von erheblicher Bedeutung. Aufgrund der notwendigen Beschränkung des Testaufwands für Komplettfahrzeuge kann beispielsweise die ausführliche Überprüfung der korrekten Funktion eines Steuergerätenetzwerkes bei den OEMs nur für einige wenige Beispielkonfigurationen durchgeführt werden. Die tatsächlich in einem individuellen Fahrzeug verbauten Soft- und Hardwareversionen sind auf den Steuergeräten angeben, und sind den Werkstätten zugänglich. Damit sind auch diese Informationen in die Diagnose des Fahrzeuges einbeziehbar. Zusätzlich wird zu Beginn einer jeden Diagnosesitzung zunächst der Verbau von Steuergeräten geprüft, falls dennoch beispielsweise von Laien oder freien Werkstätten die Soft- oder Hardware des Fahrzeugs verändert wurde.

Für die Werkstatt ist neben der Identifikation der tatsächlichen Fahrzeugausstattung die Unterteilung des Fahrzeugs in kleinste tauschbare Komponenten wichtig. Die weiterführende Fehleridentifikation auf Bauteile innerhalb dieser Komponenten ist nicht notwendig und wird aus Kostengründen auch meist nicht durchgeführt.

Diese beiden Aspekte müssen bei der Diagnose möglichst allgemein berücksichtigt werden, daher wird in den folgenden Definitionen anlehnend an [Ring99] zu diesem Zweck das Fahrzeug wie folgt unterteilt:

Definition Fahrzeug-Plattform

Eine Fahrzeug-Plattform beschreibt eine Menge von Fahrzeugen, die zu einem großen Anteil aus gleichen Komponenten aufgebaut werden.

Die Plattformstrategie ermöglicht trotz hoher Marken- und Produktvielfalt eine effiziente Entwicklung und Produktion. Im Bezug auf Volkswagen soll eine neuere Variante der Plattformstrategie, die so genannte Modulstrategie weitere Einsparungen bringen. Dabei sollen Basisbauteile wie Fahrgestelle und Antriebsstränge in kleinteilige Einzelteile, die Module, zergliedert werden, die flexibel kombiniert werden können [Pisc04]. Lediglich Teile, die das Gesicht des Fahrzeugs prägen sollen, werden differenziert entwickelt [Lang05].

Definition Fahrzeug-Typ

Ein Fahrzeug-Typ beschreibt innerhalb einer Fahrzeug-Plattform eine einzelne Baureihe, die durch ein gemeinsames Aussehen bestimmt ist.

Als Beispiel aus dem Volkswagen-Konzern kann hier der Polo als Fahrzeug-Typ der Fahrzeug-Plattform mit der Bezeichnung PQ-24 dienen. Zu dieser Plattform gehören auch die Fahrzeug-Typen Audi A2 und Seat Cordoba [Auto04].

Definition Fahrzeug-System

Ein Fahrzeug-System beschreibt eine konkrete und gültige Ausprägung eines bestimmten Fahrzeug-Typs, wie es an den Kunden ausgeliefert werden kann.

Es gibt je nach Fahrzeug-Typ eine sehr große Anzahl von möglichen Fahrzeugen. In vielen Fällen unterscheiden sich diese lediglich in der Farbe und unterschiedlichen Sitzbezügen. Aber für die hier betrachtete Diagnose von elektrischen und elektronischen Systemen finden sich relevante Unterschiede in der Anzahl und Art dieser installierten Systeme.

Definition Fahrzeug-Subsystem

Ein Subsystem ist eine Teilmenge eines Fahrzeug-Systems, das eine zusammen hängende Teilaufgabe erfüllt [DIN25424].

Die zusammenhängende Teilaufgabe wird im Allgemeinen als eine kleine Teilfunktionalität des Kraftfahrzeuges angesehen: d.h. aus dem Komfortbereich beispielsweise die Ansteuerung eines Fensters, oder im Bremsenbereich die ESP-Regelung.

Definition Fahrzeug-System in Vollausstattung

Ein Fahrzeug-System in Vollausstattung beschreibt ein Fahrzeug-System, das alle möglichen Ausstattungen eines bestimmten Fahrzeug-Typs enthält.

Dieses System ist nur ein Gedankenspiel, da es einander widersprechende Ausstattungsvarianten enthalten kann, wie z.B. eine Klimaregelungsanlage und eine Klimasteuerung.

Dieses virtuelle System kann in Bezug auf die Diagnose dazu verwendet werden, um lediglich ein sehr großes System zu betrachten. Vorteilhaft für Diagnoseanwendungen ist hierbei, ein fixes Modell betrachten zu können, während die unüberschaubare Anzahl unterschiedlicher realisierbarer Fahrzeug-Systeme für eine generelle Betrachtung zu groß wird. Sollen also statische Aussagen getroffen werden, die für alle Fahrzeug-Systeme gelten, so wird man ein Fahrzeug-System in Vollausstattung dafür heranziehen.

3.2.3.3. Elektrik und Elektronik im Fahrzeug

Generell hat die Elektrik und Elektronik eines Fahrzeugs vereinfacht eine in Abbildung 3-2 dargestellte Struktur. Die elektrischen Komponenten in Form von Sensoren und Aktoren sind über Stecker an die Steuergeräte des Fahrzeugs angeschlossen. Die Kommunikation zwischen den Steuergeräten erfolgt über Bussysteme.

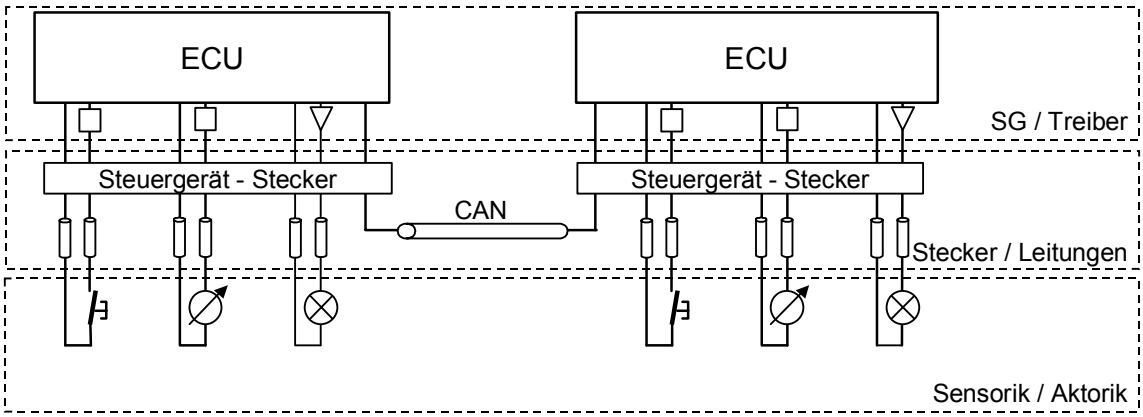


Abbildung 3-2 Schema Elektrik/Elektronik im vernetzten Kraftfahrzeug

Definition Steuergerät

Ein Steuergerät ist ein eingebettetes Mikrocontrollersystem im Automobil, das Steuerungs- und Regelungsaufgaben übernimmt.

Steuergeräte verfügen im Allgemeinen über Ein- und Ausgänge um mit dem Fahrzeug bzw. den Insassen interagieren zu können. In Abbildung 3-3 ist beispielhaft ein Fensterheber-steuergerät für die Volkswagen Plattform PQ-24 zu sehen.

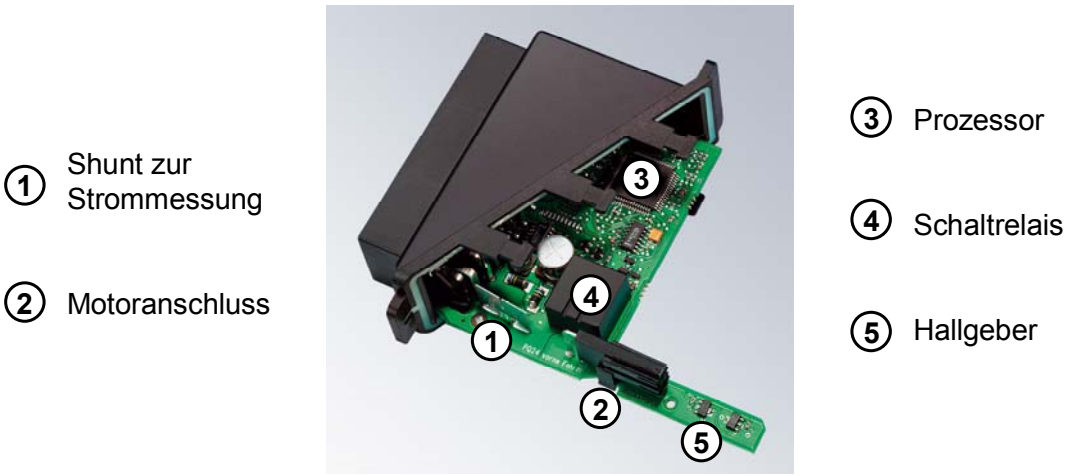


Abbildung 3-3 Fensterheber-Steuergerät

Steuergeräte können unterschiedliche Betriebsmodi annehmen. Im regulären Betrieb, d.h. wenn das Steuergerät annimmt, dass alle angeschlossene Sensorik und Aktorik sowie es selbst fehlerfrei arbeitet, befindet sich das Steuergerät im Normalzustand. Ein weiterer Betriebszustand ist der so genannte Notlauf. Er wird aktiviert, wenn das Steuergerät einen Fehler erkannt hat, der den normalen Betrieb verhindern, negativ beeinflussen oder sogar Folgefehler bewirken kann. In diesem Fall werden z.B. von Motorsteuergeräten nur Standard-Kennfelder für die Motoransteuerung verwendet, so dass der Motor zumindest läuft, wenn auch die Leistungsfähigkeit und Abgaswerte des Motors negativ beeinflusst sind. Im Bereich der Motorsteuergeräte wird dieser Modus auch als Limp-Home-Mode bezeichnet, und muss dem Fahrer angezeigt werden. Im dritten möglichen Modus, dem Diagnosemodus, ist das direkte Ansteuern von Sensorik und Aktorik sowie das Auslesen von Fehlercodes möglich z.B: über einen externen Diagnosetester. Im Diagnosemodus sind die normalen Steuerungsaufgaben teilweise deaktiviert.

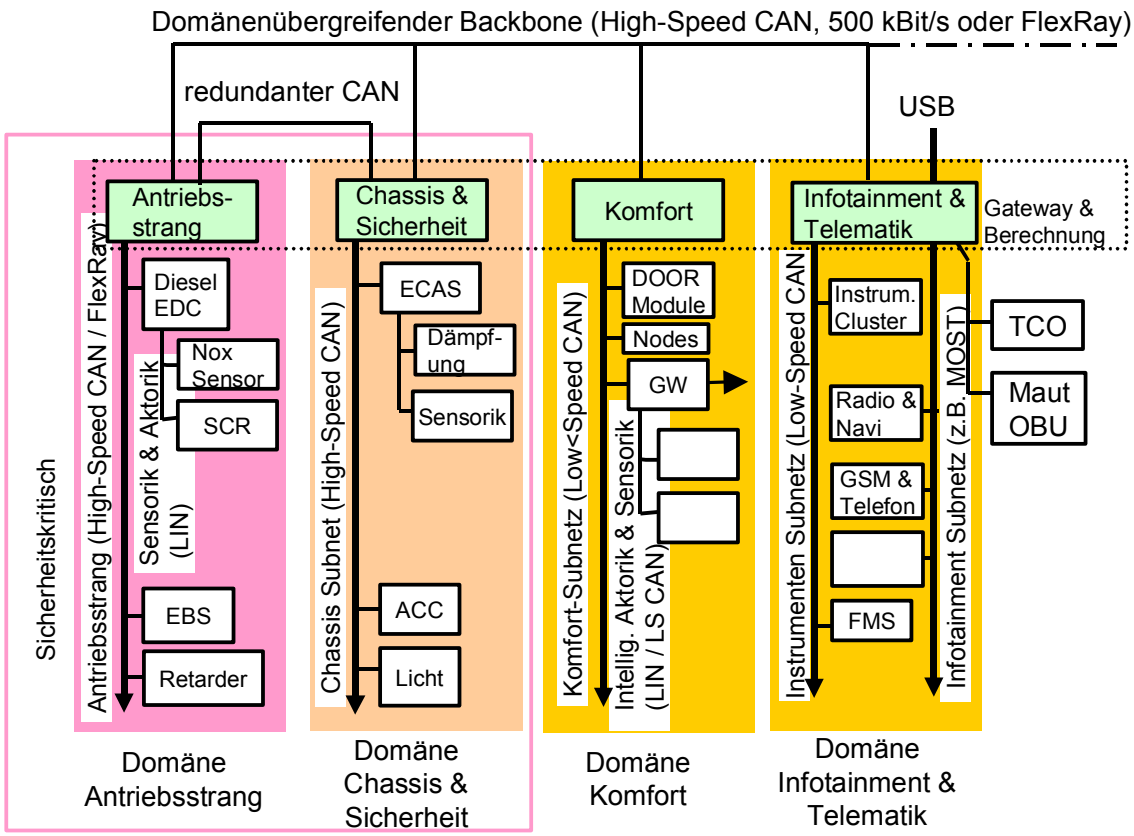


Abbildung 3-4 Schematisches Vernetzungskonzept [Felb05]

Wie in Abbildung 3-4 dargestellt, sind heutzutage die elektronischen Systeme im Automobil nicht voneinander isoliert, sondern über Bussysteme vernetzt. Die im automobilen Umfeld gebräuchlichsten Bussysteme sind das CAN, LIN, MOST und zukünftig FlexRay [Felb05].

Diese Vernetzung ermöglicht neben der Einsparung von elektrischen Leitungssträngen gegenüber einer sternförmigen Verdrahtung auch die mehrfache Nutzung von Sensorsignalen in unterschiedlichen Steuergeräten. Für die Diagnose derartiger Systeme bedeutet dies allerdings, dass am Ausfall einer einzigen Funktion mehrere Steuergeräte beteiligt sein können.

3.2.3.4. Steuergeräteeigendiagnose

Definition

> Die Steuergeräteeigendiagnose bezeichnet die Selbstüberwachung eines Steuergerätes. Dazu gehört die Überwachung von Signalen, Archivierung aufgetretener Fehler, Bereitstellung von Notlaufeigenschaften und Kommunikation mit Diagnosegeräten

Wie aus Abbildung 3-5 zu entnehmen ist, ist der Anteil der Eigendiagnose innerhalb der Steuergeräte schon seit Beginn der Einführung von elektronischen Steuergeräten in das Automobil bedeutsam und damit ein Kostenfaktor in Bezug auf zur Verfügung gestellten Permanentspeicher. Durch die restriktive Informationspolitik der Automobilhersteller und –zulieferer gibt es wenig aktuelle Vergleichsdaten. Aber auch die wenigen öffentlich verfügbaren neueren Zahlen belegen, dass die Eigendiagnose erhebliche Ressourcen benötigt. In [Wein05] werden 30% der Entwicklungskosten für die (Eigen-)diagnose angesetzt, sowie 19% des genutzten Flashspeichers von Diagnosefunktionen belegt. In diesem Beispielsteuergerät werden die Testkosten zu 25% von den Tests der Diagnosefunktionalität verursacht [Wein05].

Die Onboard-Diagnose ist im Normalfall ein passives Glied der Software des Steuergerätes. Sie hat normalerweise keinen Einfluss auf die eigentlichen Nutzfunktionen. Die Aufgaben der Steuergeräteeigendiagnose sind die Überwachung der Ein- und Ausgänge sowie offen gelegte interne Größen des Systems auf das Einhalten gültiger Bereiche und Zeiten. Bei Verletzung von gültigen Intervallen wird ein Fehlercode gesetzt, der dieses Fehlverhalten festhält und dokumentiert.

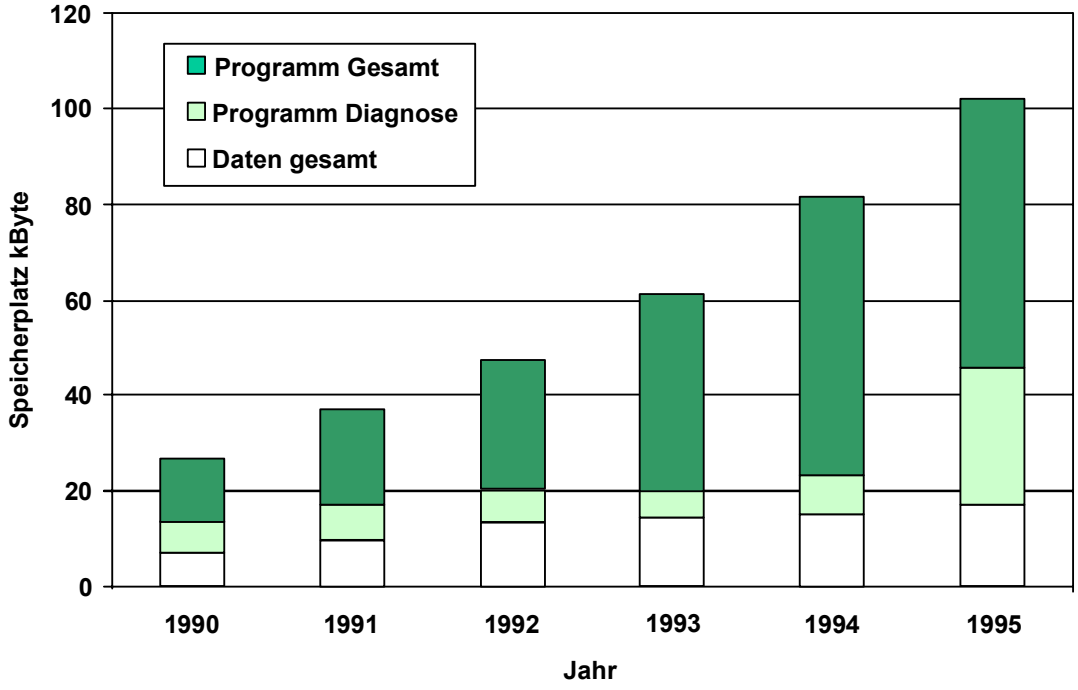


Abbildung 3-5 Anteil der Eigendiagnose am Speicherbedarf [Cosf95]

Definition Fehlercode

Ein Fehlercode dokumentiert über einen eindeutigen Bezeichner ein von einem Steuergerät erkanntes Fehlverhalten.

Dabei handelt es sich entweder um einen Fehler innerhalb des Steuergerätes oder in seinen peripheren Komponenten. Die Bedingung, wann ein bestimmter Fehlercode zu setzen ist, ist eindeutig in der Anforderungsbeschreibung des Fahrzeugherstellers dokumentiert.

Als Interaktion der Steuergeräteigendiagnose mit der Nennfunktionalität des Steuergerätes wird oft definiert, dass bei Auftreten von Fehlercodes das Steuergerät den Notlauf übergeht.

Aufbauend auf den Fehlercodes lassen sich bereits Systemdiagnosen generieren. Jedoch unterstützen Fehlerbeschreibungen und Beobachtungen, die in der Werkstatt zusätzlich in die Diagnose einfließen können, die Genauigkeit der Diagnosen.

Tatsächlich wird ein Diagnosesystem nicht immer nur auf Fehlercodes und Messgrößen der Steuergeräte basieren, sondern auch die Auswertung von anders gearteten Symptomen aufnehmen. In Anlehnung an den medizinischen Begriff des Symptoms, der eine subjektiv empfundene Beschwerde, die Auswirkung einer Krankheit oder einer Verletzung kennzeichnet [Rull01], werden analog in der Kraftfahrzeugdiagnose Beobachtungen der Werkstattmitarbeiter in die Berechnungen einbezogen, d.h. Symptome werden von den Servicemitarbeitern als

mutmaßlich in Zusammenhang mit dem Fehlverhalten des Systems stehend betrachtet und werden in die Diagnosesysteme integriert. Durch die Einbeziehung von solchen Beobachtungen können am Automobil die für die automatisierte Erkennung notwendigen, aber teuren Sensoren eingespart werden.

3.2.4. Fehlercodetypen

Die in den Steuergeräten abgelegten Fehlercodes lassen sich nach den im Folgenden beschriebenen unterschiedlichen Typen klassifizieren. Die entsprechende Eingruppierung der Fehlertypen mitsamt der Definition, anhand welcher Daten sie erkannt werden sollen, findet sich in der Spezifikation des Fahrzeugherstellers.

3.2.4.1. Signal unplausibel

Es wird überprüft, ob ein Signal einen in der Anforderungsbeschreibung definierten Gültigkeits-Bereich oder Verlauf einhält. Beim Verlassen der zuvor definierten Intervallgrenzen wird der Fehlercode gesetzt. Je nach Signaltyp und Anforderung wird der Fehlercode erst gesetzt, wenn der Fehlerzustand eine bestimmte Zeit vorgelegen hat. Zusätzlich lassen sich mit diesem Fehlercode Sensorsignale als unplausibel erklären, wenn das Signal anderer Sensoren dem aktuellen Messwert widerspricht, oder aktuelle Messwerte nicht mit der Signalhistorie in sinnvollen Einklang zu bringen sind. Beispielsweise liegt ein unplausibles Signal vor, wenn sich die Öltemperatur sprunghaft ändert.

3.2.4.2. Signal zu niedrig, Signal zu hoch

Ähnlich wie bei der Plausibilitätsprüfung wird durch diesen Fehlercode das Verlassen eines Gültigkeitsbereichs angezeigt. Der Spezialfall hier kennzeichnet jedoch das absolute Über-, bzw. Unterschreiten von Gültigkeitsgrenzen. Für Strommessungen beispielsweise kann „Signal zu hoch“ einen Kurzschluss zur Masse anzeigen, während „Signal unplausibel“ lediglich anzeigt, dass der Strom in einer Größe fließt, die so nicht erwartet wurde.

3.2.4.3. Kommunikation gestört

Stellt ein Steuergerät fest, dass es im Betrieb den Kontakt zu den anderen bzw. einem bestimmten Steuergerät verloren hat, so wird dieser Fehlercode gesetzt. Als Fehlerursachen können Wackelkontakte bzw. Leitungsprobleme in dem Busanschluss der betroffenen Steuergeräte sein, oder das nicht mehr kommunizierende Steuergerät ist „abgestürzt“. Die dafür möglichen Ursachen sind EMV-, Software- oder generelle Busauslastungs-Probleme.

3.2.4.4. Anwendungsspezifische Fehlercodes

Zusätzlich können weitere Fehlercodetypen definiert sein, die auf die spezielle Anwendung zugeschnitten sind. Ein Beispiel für die Anwendbarkeit eines solchen Fehlercodes ist die Überwachung eines Relais: „Relais schließt bzw. öffnet nicht“

3.2.5. Fehlerarten

Für die Auswertung von Fehlercodes ist die Dauer des Anliegens eines Fehlerbildes von Bedeutung. Abhängig davon, ob ein ermittelter Fehler zum Zeitpunkt der Diagnose noch besteht oder nicht, unterteilt man in sporadische Fehler und statische Fehler.

3.2.5.1. Sporadische Fehler

Ein sporadischer Fehler bezeichnet einen Fehler, der mindestens einmal aufgetreten ist, aber zum Zeitpunkt der Diagnose nicht mehr vorliegt. Erklärungen für sporadische Fehler können Wackelkontakte oder thermische Probleme sein. In der Praxis werden in den Steuergeräten aufgrund der begrenzten Speicherkapazitäten für Fehlercodes Zähler integriert, die sporadische Fehler nach einer gewissen Anzahl von fehlerlosen Fahrzyklen wieder vollständig löschen. Ebenso können aktuell nicht mehr bestehende Fehler bei einem vollen Fehlerspeicher durch aktuell anliegende Fehlerzustände komplett überschrieben und damit gelöscht werden. In diesem Fall fehlt der Diagnose in der Werkstatt die Information, dass dieser Fehlercode bereits einmal gesetzt wurde.

Für die Diagnose ist es besonders für sporadisch auftretende Fehler wichtig, dass Umgebungsbedingungen, d.h. Temperaturen, Kilometerstände etc. beim ersten Auftreten im sogenannten Freeze-Frame gespeichert werden. Aus diesen Daten lassen sich Rückschlüsse hinsichtlich der tatsächlichen Fehlerursache finden [ISO14230][ISO15765].

3.2.5.2. Statische Fehler

Als statischen Fehler werden Zustände charakterisiert, die nach dem ersten Auftreten kontinuierlich weiter bestehen. Diese Art von Fehlern ist für die Diagnose leichter zu identifizieren, da keine besonderen Randbedingungen bestehen müssen, damit sich der fehlerhafte Zustand einstellt. Der entsprechende Fehlercode ist und bleibt gesetzt, und kann in der Diagnosemaschine verarbeitet werden. Die Bedingungen, die zum Setzen des Fehlercodes geführt haben, sind ständig messbar und können damit in der Werkstatt überprüft werden [ISO14230].

3.2.5.3. Abhängige und unabhängige Fehler

In der diagnostischen Betrachtung von Fehlern wird zwischen unabhängigen und voneinander abhängigen Fehlern unterschieden. Mehrfache unabhängige Fehler in einem System zur gleichen Zeit sind sehr unwahrscheinlich, weil sich die gleichzeitige Auftretenswahrscheinlichkeit aus dem Produkt der unabhängigen Auftrittswahrscheinlichkeiten ergibt. Daher erklären die meisten Diagnosesysteme Fehlerbilder mit vielen betroffenen Komponenten vorrangig durch voneinander abhängige Fehler.

3.2.5.4. Primäre Fehler

Ein primärer Fehler, auch Einfachfehler genannt, ist derjenige Fehler, der das Vorliegen aller Fehlerbilder erklärt. Dieser Fehler ist auch die Ursache für alle von ihm abhängigen sekundären Fehler, wie z. B. durchgebrannte Sicherungen etc.

Laut [DIN25424] werden die Einfachfehler auch als primäre Fehler bezeichnet, da sie nur bei zulässigen Einsatzbedingungen auftreten, d. h. zum Zeitpunkt des Eintretens des Defekts sind aus dem Blickwinkel der defekten Komponente betrachtet alle weiteren mit ihr verbundenen Komponenten als fehlerfrei anzusehen.

3.2.5.5. Sekundäre Fehler

Sekundäre Fehler bezeichnen Fehlzustände, die aufgrund des Vorliegen eines primären Fehlers auftreten. Dabei kann es sich beispielsweise um das Durchbrennen einer Sicherung infolge eines Primärfehlers wie einem Masseschluss handeln. Durch den Primärfehler funktionieren andere, sonst unabhängige Systemteile nicht mehr spezifikationsgemäß. Das reine Ersetzen der Sicherung führt nicht zum Beheben der Störung, da die neue Sicherung ebenfalls durchbrennen wird, solange der Primärfehler nicht behoben ist.

3.2.6. Diagnosetiefe

Die Diagnosetiefe beschreibt, welche Granularität die kleinste diagnostizierbare Komponente des Systems hat. In Abbildung 3-6 ist anhand eines Beispielsystems dargestellt, welche Ausgaben ein Diagnosesystem mit unterschiedlicher Diagnosetiefe erzeugen könnte. Im Falle geringer Mengen an verfügbaren Daten könnte der Diagnosetechniker lediglich dahin geführt werden, zu analysieren, ob die einzelnen Teilsysteme funktionieren. Bei Defekten innerhalb eines Teilsystems müsste das gesamte Teilsystem ausgetauscht werden, wenn sich der Techniker damit nicht genau auskennt, und den eigentlichen Fehler unabhängig vom Diagnosesys-

tem identifizieren kann. Im Falle der großen Diagnosetiefe werden die notwendigen Einzelschritte, den Fehler exakt zu finden dem Techniker Schritt für Schritt vorgegeben.

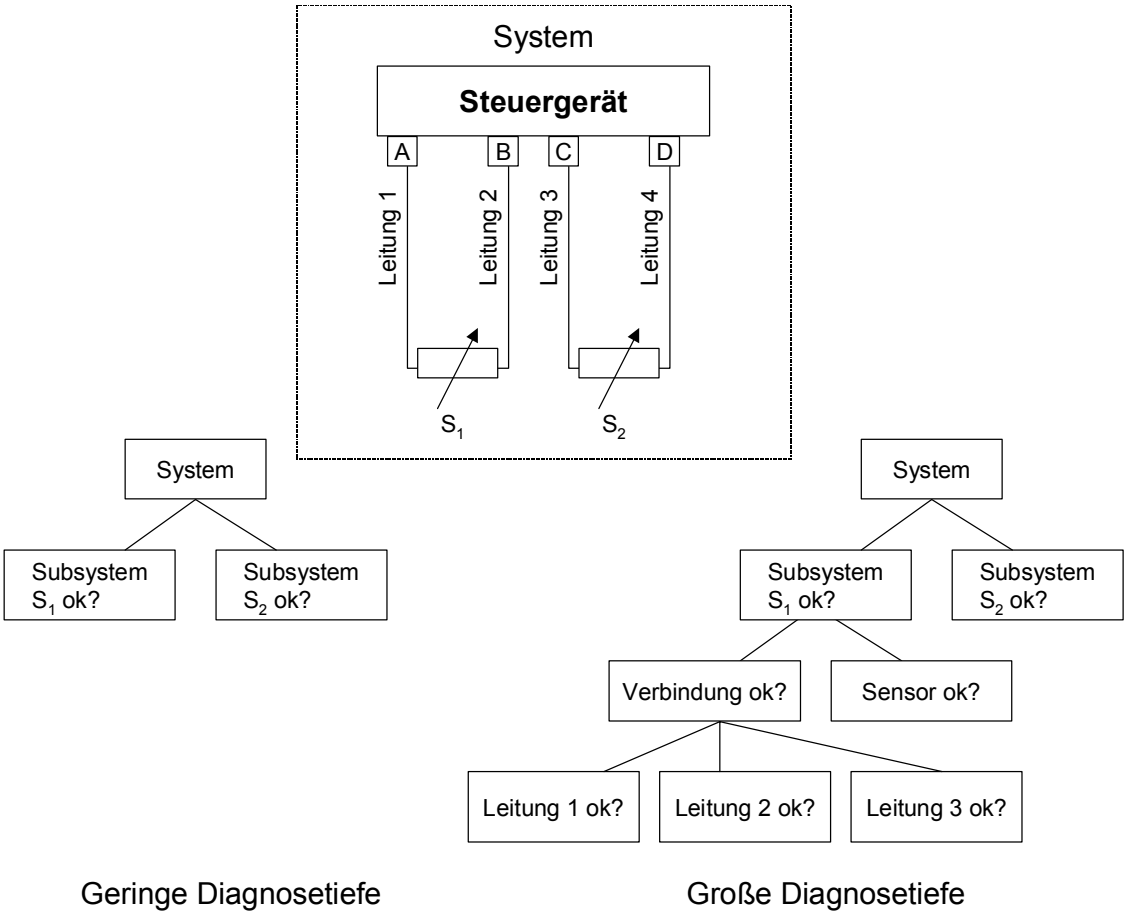


Abbildung 3-6 Abwägung der Diagnosetiefe eines Diagnosesystems [Ring99]

Geringe Diagnosetiefen können dem Diagnosetechniker lediglich Subsysteme nennen, in denen der Fehler vermutet wird, während größere Diagnosetiefen die Einschränkung bis auf Komponentenebene erlauben. Damit gewähren geringe Diagnosetiefen dem Diagnosetechniker mehr eigenen Spielraum zum Identifizieren des Fehlers, verlangen aber auch hoch qualifiziertes Personal. Die große Diagnosetiefe resultiert in genauen Anweisungen, welche Komponenten potentiell fehlerhaft sind, und gibt vor, in welcher Reihenfolge weitere Prüfschritte durchzuführen sind, um den Fehler weiter einzugrenzen.

Prozedurale Diagnoseverfahren kommen im Allgemeinen nur auf geringe Diagnosetiefen, da mit vertretbarem Aufwand lediglich eine kleine Anzahl von Diagnosefällen in dem System berücksichtigt werden kann. Eine allgemeine Aussage über die richtige Dimensionierung der Diagnosetiefe lässt sich nicht treffen. Dies hängt von dem verwendeten Diagnoseverfahren

und den für die Diagnose zur Verfügung stehenden Ressourcen in Form von Steuergeräteressourcen und Entwicklungskosten ab.

Ist eine große Diagnosetiefe gefordert, so bedeutet dies unter Umständen für den Steuergerätelieferanten den Verbau von zusätzlicher Sensorik, um die Diagnose auch präzise genug durchführen zu können. An dieser Stelle muss ein Gleichgewicht zwischen technischer Machbarkeit und Praktikabilität gefunden werden. Dabei muss entweder das Hardwarelayout an die geforderte Diagnosetiefe angepasst werden, indem zusätzliche Sensorik verbaut wird, oder es kann nur eine allgemeine Aussage getroffen werden.

Bei der Erstellung des Diagnosesystems ist ebenfalls zu beachten, dass die Diagnosetiefe auch an die Kenntnisse des Werkstattpersonals angepasst werden sollte. Zu detaillierte Prüfungen mit nicht erkennbarem Hintergrund der Fragestellung werden vom Werkstattpersonal nicht akzeptiert und verringern damit die Akzeptanz des Diagnosesystems [Raun02].

3.2.7. Zusammenfassung

Im Folgenden sind die wichtigsten Eigenschaften zusammenfassend dargestellt, die für die Auswahl eines geeigneten Diagnosealgorithmus beachtet werden müssen:

- Die Elektrik und Elektronik eines Fahrzeugs kann nicht unabhängig von der Mechanik betrachtet werden, in die sie eingebettet ist. Es handelt sich um mechatronische Systeme, die untereinander vernetzt sind.
- Die eingesetzten Steuergeräte interagieren über Kommunikationssysteme, Sensoren und Aktoren mit der Umwelt.
- Die Anzahl von Sensoren bzw. Aktoren ist im Vergleich zu der Anzahl der im System verbauten elektrischen und mechanischen Elemente sehr klein. Eine einhundertprozentige sensorische Abdeckung aller Systempunkte ist aus Kostengründen nicht realisierbar.
- Aus den Steuergeräten lassen sich Eingangsinformation für ein Diagnosesystem auslesen: aufgetretene Fehlercodes, aktuelle Sensorwerte und aktuelle Aktoransteuerungen.
- Die Aktoransteuerung kann im Rahmen von Stellgliedtests direkt erfolgen und damit überprüft werden.
- Die Kommunikation zwischen Steuergeräten und dem Diagnosesystem erfolgt über eine standardisierte Verbindung, in älteren Fahrzeugen über die K-Leitung, in neueren über den CAN-Bus. Für zukünftige Fahrzeuge ist auch ein drahtloser Zugang in Planung.

- Es gibt sehr viele Ausprägungen innerhalb eines Fahrzeugtyps. Die unterschiedlichen Varianten können sich aus Sicht der Diagnose in ihrem strukturellen Aufbau, der Anzahl und Art der Steuergeräte, sowie in Hardware- und Softwareversionen unterscheiden. Der tatsächliche Systemaufbau ist beim Einfahren in die Werkstatt nicht bekannt.
- Fehlercodeinformationen sind Bestandteil von Beobachtungen, die die Steuergeräte vor dem Zeitpunkt getätigt haben, an dem das Gesamtsystem in der Werkstatt überprüft wird. Daher können unter bestimmten Umständen die Ursachen für die erkannten Fehler zum Zeitpunkt des Werkstattbesuchs nicht mehr vorliegen.
- Es können sowohl systeminterne Messpunkte, die von systemeigenen Sensoren überwacht werden, als auch menschliche Beobachtungen und externe Prüfungen in die Diagnose einbezogen werden.
- Die Diagnosetiefe beschreibt die Granularität der Fehleraussagen des Diagnosesystems. Die geforderte Diagnosetiefe, muss mit vertretbarem Aufwand über alle Fahrzeugvarianten erreichbar sein.

4. Stand der Technik der Diagnose

An dieser Stelle soll ein Überblick über die zur Zeit der Erstellung der Arbeit bekannten Diagnoseverfahren gegeben werden. Zum Schluss wird ein anhand der ermittelten Informationen über die Diagnoseverfahren ein für die vorliegende Arbeit geeignetes Verfahren ausgewählt.

4.1. Konventionelle Werkstattsysteme und eingesetzte Diagnoseverfahren

Im Folgenden sollen die bisher vorliegenden Lösungen zur Diagnose im Automobil aufgeführt und jeweils kurz erläutert werden, um die Basis der Arbeit für ein neues Diagnosesystem zu bestimmen.

4.1.1. Inbetriebnahme

Bereits in der Produktion liefert die konventionelle Steuergeräteigendiagnose Hinweise auf während des Herstellungs- und Inbetriebnahmeprozesses entstandene Defekte. Zur Sicherstellung der Produktqualität werden daher die Einzelsysteme direkt vor dem Einbau auf Fehlerfreiheit getestet. Nach dem Einbau wird nach einem Probelauf des Kraftfahrzeugs der Fehlerspeicher aller Steuergeräte ausgelesen und damit der Verbund der Steuergeräte auf entstandene Fehler untersucht. Zur Fehlerbehebung werden hier die Original-Werkstattester der OEMs verwendet, deren nähere Eigenschaften im Folgenden Abschnitt beschrieben werden.

4.1.2. Werkstattsysteme

Vielfach im Einsatz sind bereits seit langem Systeme wie in Abbildung 4-1, diese realisieren auf einem ein- bis mehrzeiligen LC-Display lediglich die Standard-OBD-Diagnose. Damit können demnach lediglich die aktuellen Motorbetriebswerte dargestellt, Stellgliedtests vorgenommen und die Fehlercodes ausgelesen werden. Bei dieser Form der Diagnosegeräte ist das Werkstattpersonal gefordert, anhand seines Fachwissens die eigentliche Fehlerdiagnose selbst vorzunehmen [Mega05].

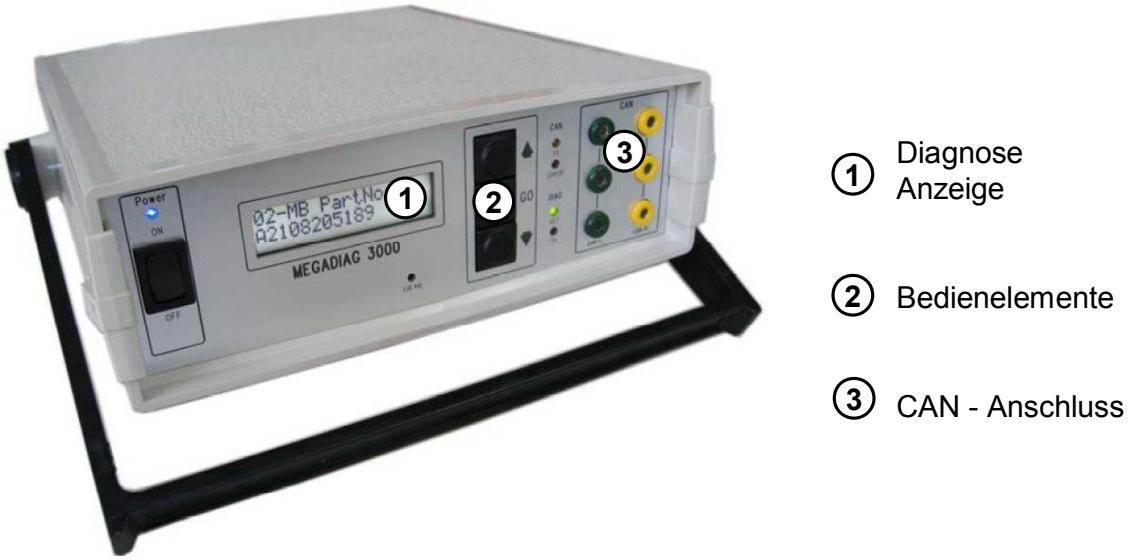


Abbildung 4-1 Diagnosegerät MEGADIAG 3000 Foto: Megacomm

In den Werkstätten sind aber vor allem in den letzten Jahren weiter entwickelte Systeme im Einsatz, wie in Abbildung 4-2 dargestellt. Sie basieren auf einer PC-Plattform mit Bildschirm, Drucker, OBD-Schnittstelle zum Fahrzeug und Messdatenerfassungsmöglichkeiten. Per CD-ROM oder online über das Internet werden die Software und die Fehlersuchbäume auf dem aktuellen Stand gehalten.

Abbildung 4-2 VAG Tester VAS 5051B Foto: Siemens

Aufgrund des großen Speicherangebots auf PC-Systemen sind hier meist zu allen Arbeitsschritten die notwendigen Serviceanweisungen mit Bildern integriert. Für Werkzeuge mit geführter Fehlersuche wird auf Expertenwissen zurückgegriffen, die zugehörenden Diagnoseprogramme werden für jede zu prüfende Funktion speziell implementiert und parametriert. Als Grundlage für die Experten dienen Erfahrungen aus bisherigen Reparaturen und a priori Überlegungen, welche Fehler durch das Auftreten bestimmter Fehlercodes erklärt werden könnten. Durch die umfangreiche Anzahl von möglichen Fahrzeugkonfigurationen und Systemausstattungen sind diese Systeme aber im praktischen Einsatz nicht besonders zuverlässig und zielführend. Der Fehlersuchbaum endet oft schnell in der Anweisung, das Steuergerät mit dem Fehlereintrag zu tauschen [Bosc03][Würt05].

Nach Angaben bei einer Anfrage in einer Kundendienstwerkstatt zum praktischen Umgang mit den eingesetzten Diagnosesystemen, wird im Zweifelsfall das Steuergerät getauscht bzw. die Fehlerergebnisse per Mail zum Service-Center des entsprechenden Fahrzeugherstellers geschickt. Der Werkstattmitarbeiter wird anhand des Diagnosegerätes geführt und kann entsprechende Maßnahmen einleiten. Bei offenen Fragen, bzw. solchen, die nicht in der Diagnoseführung enthalten sein sollten , muss er sich mit dem Fahrzeughersteller in Verbindung setzen. Die getauschten und ersetzten Teile werden Volkswagen über die entsprechenden Abrechnungen und Bestellungen zugänglich gemacht. Die im Verlauf der Diagnosesitzungen ermittelten Daten und Abläufe an die Fahrzeughersteller übermittelt, um spätere Analysen der Diagnosequalität und die Überprüfung der Abrechnung von Gewährleistungskosten zu ermöglichen.

4.1.3. Prozedurale Diagnose

Prozedurale Diagnoseverfahren verschmelzen Domänenwissen und Inferenzmechanismus. Der Diagnoseautor erstellt für ein vorgegebenes zu diagnostizierendes System eine Folge von Prüfschritten, die die möglichen Fehler eingrenzen sollen. Typischerweise werden die Abfolgen der durchzuführenden Prüfschritte anhand eines Entscheidungsbaums vorgegeben. Ausgabe eines prozeduralen Diagnosesystems ist demnach eine Handlungsanweisung an einen Servicetechniker, aufgrund dessen Erkenntnissen die Diagnose weiter durchgeführt werden kann. Problematisch ist hier einerseits der Ausschluss des Servicemitarbeiter aus der Schlussfolgerungskette, da er nicht zwingend in Kenntnis darüber gesetzt wird, welche Fehlerhypothese mit einem bestimmten Prüfschritt überprüft werden soll. Andererseits ist die Erstellung der notwendigen Fehlersuchbäume nur für einfache Systeme manuell zu bewältigen. Für

komplexere Systeme existieren Ansätze, die Entscheidungsbäume mit Rechnerhilfe optimal zu gestalten [Oliv03].

4.1.4. Regelbasierte Diagnose

Regelbasierte Diagnosesysteme sind dadurch charakterisiert, dass zum Einen das Diagnosewissen von den Inferenzmechanismen getrennt verwaltet wird, und zum Anderen, dass sie durch das vorgegebene Expertenwissen in der Lage sind, die Ursachen eines beobachteten Fehlverhaltens zu bestimmen. Dieses Wissen wird durch Expertenbefragung gewonnen und besteht aus Symptom-Fehler-Beziehungen in Form von Regeln. Ein Beispiel für ein angewandtes wissensbasiertes Diagnosesystem ist die Hilfe bei Problemstellungen mit Treibern im Betriebssystem Microsoft Windows. Hierbei wird der Benutzer anhand eines starr vorgegebenen Ablaufs mittels Fehlersuchbäumen zur Diagnose seines Problems und anschließenden Fehlerbehebung angeleitet. Dabei werden die begrenzten Möglichkeiten eines solchen Systems deutlich:

- Beschränkung auf bekannte Symptome und Fehler:

  Es können lediglich Symptome und Fehler behandelt werden, die in das System von den Experten eingegeben wurde. In vielen Fällen ist das nicht ausreichend, da eine komplette Liste von Symptom-Fehler-Beziehungen nur bei einfachen oder sehr gut bekannten Systemen erstellt werden kann.

- Fehlerbeeinflussung:

  Mehrfachfehler können Symptome eines darin enthaltenen Einzelfehlers überdecken. Mehrfachfehler sind aufgrund der unüberschaubaren Menge an möglichen Auftrittskombinationen in solchen Systemen im Allgemeinen nicht diagnostizierbar.

- Beschränkte Allgemeingültigkeit:

  Das Diagnosesystem ist nur anwendbar auf das System, das auch den Experten vorlag. Bereits kleine Modifikationen an der ursprünglichen Konfiguration, wie sie vor allem durch die flexibel zu wählende Ausstattungsvarianten im Automobil zu finden sind, können zu Fehlinterpretationen durch das System führen.

- Keine Zusammensetzbarkeit:

  Die Systemstruktur ist nur implizit durch die Symptom-Fehlerbeziehungen gegeben, d.h. es gibt keine Regeln für das Erstellen eines Diagnosesystems. Das heißt auch, dass die

Anpassung eines gegebenen Systems auf ein ähnliches System mit z.B. leicht veränderter Struktur nicht systematisch vorgenommen werden kann.

Diese Einschränkungen führen dazu, dass diese Art der Erstellung von Diagnosesystemen nur in begrenzten Bereichen verbreitet ist. Vor allem die immer kürzeren Produktzyklen und Markteinführungszeiten bei steigender Komplexität der Fahrzeugsysteme erlauben es nicht, mit vertretbarem Aufwand an Kosten genügend Expertenwissen in ein wissensbasiertes Diagnosesystem zu integrieren. Die Entwicklungs- und Wartungskosten für die händisch erstellte Diagnose eines komplexen Fahrzeugsystems mit kann daher leicht die für Diagnosesysteme akzeptablen Kosten übertreffen.

4.1.5. Modellbasierte Diagnose

Die modellbasierte Diagnose verwendet für Aussagen zum Zustand eines Systems sowohl ein Modell des Verhaltens als auch ein Strukturmodell. Dabei wird wie in wie in Abbildung 4-3 dargestellt, versucht, Messdaten des tatsächlichen vernetzten Systems mit dem berechneten Verhalten eines Modellsystems in Einklang zu bringen.

Dazu werden die Komponenten des Diagnosemodells so ausgelegt, dass sie je nach Verfahren sowohl das korrekte als auch das fehlerhafte Verhalten der realen Komponenten abbilden. Ebenso können teilweise auch Strukturveränderungen des Systemmodells beim Vorliegen bestimmter Fehler als Fehlermodelle hinterlegt werden. Das Diagnoseverfahren minimiert dann durch Modifikation des Systemmodells mit Hilfe der hinterlegten OK- bzw. Fehlermodelle, die Differenz zwischen den Ausgaben der Modelldaten und der Antworten des realen Systems. Wenn sich die Ausgaben des Modellsystems mit den tatsächlichen Antworten des realen Systems decken, gilt der Fehler als gefunden und kann ausgegeben werden.

Die Güte eines solchen Verfahrens ist begrenzt durch die Güte der hinterlegten Fehlermodelle der Komponenten, sowie die Genauigkeit der möglichen Strukturveränderungen des Systemmodells durch Fehler. Je nach Komplexität und Zerlegbarkeit des Systems wird es schwierig, einzelne Systemkomponenten zu identifizieren, und deren Fehlverhalten exakt zu hinterlegen. Je nach Anforderung an die Veränderungsgeschwindigkeit des Prozesses kann hier der Abgleich der Systemdaten mit den Modelldaten nicht in Echtzeit durchgeführt werden, so dass ein solches Vorgehen entweder eine gute Messwertverarbeitung benötigt, die nur eine begrenzte Zahl von Systemveränderungen an das Diagnosesystem weitergibt, oder hochperformante Rechnersysteme.

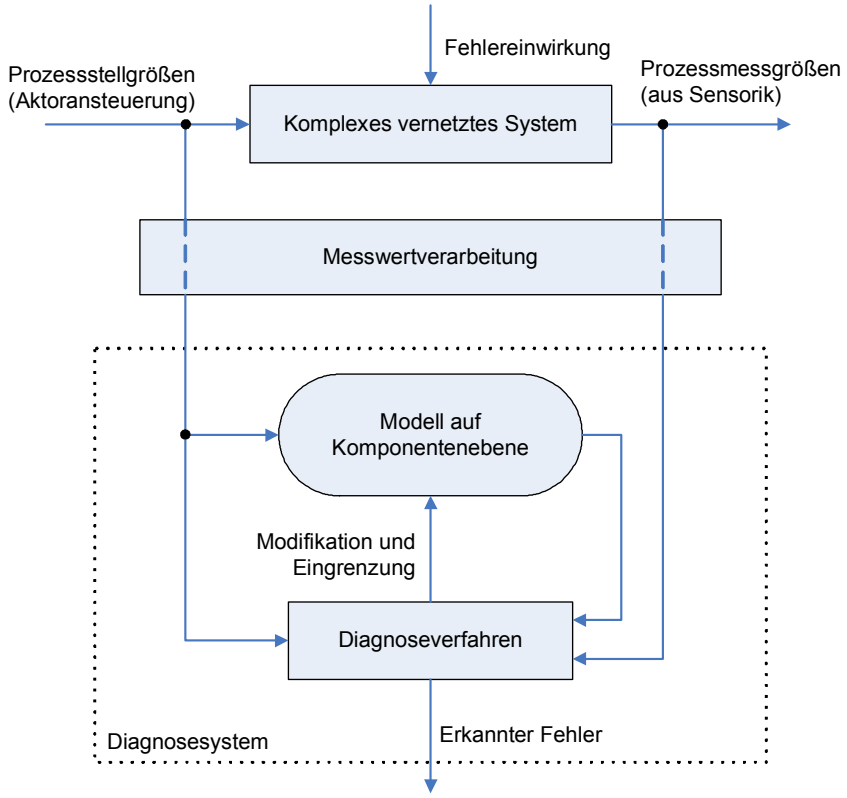


Abbildung 4-3 Struktur der modellbasierten Diagnose [Hein99]

Wie oben beschrieben sind wissens- bzw. regelbasierte Systeme sehr schnell bei der Berechnung von Fehlerkandidaten, da die zugrunde liegenden Folge von Schlüssen in sehr effizienter Weise z.B. in Entscheidungsbäumen abgearbeitet werden kann.

Nachteilig am Einsatz wissensbasierter Diagnose ist der Aufwand für die Erzeugung der Wissensbasis. Diese kann durch Expertenwissen geschaffen werden, indem Experten versuchen, alle kritischen Fälle zu beschreiben und für diese Fehlercodes zu definieren. Auf diese Weise werden leicht viele Diagnosefälle übersehen und der Komplexität des Systems wird nur unzureichend Rechnung getragen.

Alternativ oder ergänzend ist es daher bei modellbasierten Systemen möglich, die Wissensbasis automatisiert aus dem Modell zu erzeugen. Problematisch ist hier der Rechenaufwand bei der Erstellung, und der schnell explodierende Zustandsraum der Diagnoseregeln, der auf den Steuergeräten fest gespeichert werden muss.

Die Verwendung von Diagnosealgorithmen wie z.B. der unten beschriebenen General Diagnosis Engine (GDE) erfordert allerdings auf dem Steuergerät erhebliche Ressourcen in Bezug auf Speicherplatz und Rechenleistung. Der Kompromiss zwischen den Kosten für die an Bord des Automobils benötigte Rechenleistung und der zur Erstellung der Diagnosemodelle

benötigten Rechenleistung bei der Entwicklung ist im Vorhinein zu treffen. Dementsprechend fällt auch die Entscheidung für die regelbasierte oder die modellbasierte Diagnose.

4.1.5.1. General Diagnosis Engine (GDE)

Grundlage der modellbasierten Diagnose für die Berechnung von Fehlerkandidaten ist die General Diagnosis Engine (GDE) [DeKl87] die auf dem Assumption based Truth Maintenance System (ATMS) [DeKl86] basiert.

Ansatz der GDE ist die Konsistenzprüfung eines Systems auf der Grundlage von Annahmen. Das Modell wird dabei in einer Reihe von Systemgleichungen beschrieben, die sich aus der Systemstruktur und dem Verhalten der Systemelemente ergeben. Die Beobachtungen des Systems über seine Umwelt gehen als Annahmen in die Berechnungen des GDE-Algorithmus ein.

Für jede berechnete Systemgröße wird während der Berechnung die Information mitgeführt, welche Annahmen bzw. Beobachtungen zu ihrer Bestimmung verwendet wurden. Im Normalfall ist das System konsistent, so dass sich bei der Berechnung der Systemgrößen durch die verschiedenen Beobachtungen keine Widersprüche ergeben. Treten jedoch Fehler in dem System auf, ergeben sich einander widersprechende Annahmen für die Systemgrößen. Der kleinste gemeinsame Satz von Annahmen, der zu der Inkompatibilität führt, wird als Diagnose-Hypothese angenommen [DeKl86][DeKl87]. Durch die Anwendung einer Datenbank können alle bisher betrachteten Fälle dort abgelegt werden und brauchen beim neuen Auftreten der Situation nicht neu berechnet zu werden. Erkauft wird dieser Vorteil durch den großen Speicherbedarf, den dieses Verfahren benötigt. Der Bedarf an RAM-Speicher steigt exponentiell mit der Zahl der Eingangsgrößen des Systems und der Anzahl ihrer möglichen Zustände an. Damit ist dieses Verfahren bereits bei recht einfachen Systemen nicht mehr auf einem eingebetteten System ablauffähig, das unter dem in der Automobil-Branche existierendem Kostendruck eingesetzt wird. Zur Begegnung des Problems des exponentiellen Wachstums der in der Wissensbasis abzulegenden Diagnoseinformationen ist die Vergabe von Wahrscheinlichkeiten für Komponentenfehler möglich. Damit kann die ATMS auf wahrscheinliche Fehlerkandidaten fokussiert werden [ROSE03]. Aber auch alternative Lösungen sind möglich, indem die zugrunde liegenden Mechanismen der GDE anhand einer optimierten Implementierung des Algorithmus auf ein superpolynominales Problem zurückgeführt werden [Fija02].

Die in STEP-X in der ersten Phase eingesetzten Entwicklungswerkzeuge zur Berechnung von Fehlerkandidaten basieren auf Weiterentwicklungen der GDE, die auch das Fehlerverhalten

von Komponenten in die Berechnungen einbeziehen. Mit Hilfe der zusätzlichen expliziten Einführung von Fehlermodi kann eine Fehleridentifikation erreicht werden. Bei nicht zutreffenden bzw. unzureichenden Fehlermodellen werden unbekannte Fehler angenommen [Stru89].

4.1.5.2. Probabilistische Netzwerke

Probabilistische Wissensrepräsentation wird vorwiegend in den heutigen Expertensystemen der künstlichen Intelligenz genutzt. Dabei wird Wissen über eine eng umrissene Domäne mit Hilfe einer Wahrscheinlichkeitsverteilung der gegenseitigen Abhängigkeiten angegeben. Bei einer komplexen Domäne, wie sie bei der Diagnose im Automobil gegeben ist, wird die Verteilungsfunktion vieldimensional und die Erfassung und Handhabung der Abhängigkeiten bereitet Schwierigkeiten.

Probabilistische Netzwerke basieren auf der Idee, den hochdimensionalen Darstellungsraum der Wahrscheinlichkeiten in mehrere niedrigdimensionale und ggf. überlappende Teildomänen aufzuteilen. Es wird auch oft von grafischen Modellen gesprochen, wobei „grafisch“ bedeutet, dass die entsprechenden Modelle gerichteten Graphen im Sinne der Graphentheorie entsprechen [Görz00].

Die Dimensionen im Zusammenhang mit probabilistischen Netzen beschreiben einen Zustand bzw. ein Objekt durch eine Menge von Attributen. Für die Beschreibung eines Automobils werden beispielsweise die Attribute Modell, Farbe und Sonderausstattung angegeben. Jedes Attribut mit seiner gültigen Wertemenge bildet eine Dimension des aufgespannten Raumes.

Ein spezifisches Fahrzeug entspricht somit einem Punkt in diesem Raum [Görz00].

In einer Verteilung δ wird jedem Punkt des Raumes eine Wahrscheinlichkeit zugeordnet, der die A-priori-Wahrscheinlichkeit angibt, dass der entsprechende Zustand des modellierten Weltausschnitts vorliegt. Die dafür notwendigen Daten können aus Produktionsdaten abgeleitet und um Expertenschätzungen ergänzt werden [Görz00].

Die große Anzahl der Punkte in einem hochdimensionalen Raum schließt aus, dass dieses Vorgehen für reale Automobile angewendet werden kann, daher wird die Verteilung δ in niederdimensionale Teilverteilungen $\{\delta_1, .. , \delta_s\}$ zerlegt und kann aus diesen auch wieder rekonstruiert werden [Görz00].

Die Idee der probabilistischen Netzwerke ist aber, ohne Rekonstruktion der Gesamtverteilung, allein basierend auf den Teilverteilungen Schlussfolgerungen zu ermöglichen. Dabei werden

die Informationen von Unterraum zu Unterraum weitergegeben. Dieses Vorgehen wird auch als Evidenzpropagation bezeichnet [Görz00].

Wichtige Anwendungen im Bereich der probabilistischen Netze sind die Markov-Netze als auch die Bayes-Netze. Für Diagnosezwecke im Automobil wird momentan der Einsatz von Bayes-Netzwerken geprüft, deshalb werden diese im Folgenden detaillierter beschrieben [Luek04].

4.1.5.2.1. Bayes Netze

Bayes Netze sind benannt nach ihrem Erfinder Thomas Bayes (1702-1761), einem Priester, der sich in einem Essay mit bedingten Wahrscheinlichkeiten auseinander setzte [Pric63]. Tatsächliche Anwendungen ergaben sich erst gegen Ende der 1980er Jahre, als genügend Rechenleistung und Speicher zur Bewältigung der notwendigen Rechenoperationen zur Verfügung stand. Haupteinsatzgebiete sind heute die medizinische Diagnose und Spam-Filter für E-Mails. Es gibt aber auch bereits Anwendungen im Bereich der Automobiltechnik [Borg04].

Bayes-Netze sind eine Menge von Zufallsvariablen und gerichteten Kanten, die zusammen einen gerichteten azyklischen Graphen ergeben. Für die Benennung der Knoten wird eingeführt, dass ein Knoten, der mit einer gerichteten Kante von X nach Y führt, als Elternknoten von Y gilt [parent(Y)]. Zusätzlich besitzt jeder Knoten X_i für alle seine Elternknoten eine Tabelle bedingter Wahrscheinlichkeiten, die die Wirkung bzw. den Effekt der Elternknoten auf X_i quantifizieren [Habe05].

Zur Verdeutlichung dient ein bekanntes Beispiel über die Funktion einer Alarmanlage [Russ03]. Der Vorreiter bei der Anwendung in Bayes-Netzwerken Judeas Pearl sagt hierbei: „Ich bin bei der Arbeit, als mein Nachbar John anruft, dass meine Alarmanlage schrillt. Aber meine Nachbarin Mary ruft nicht wegen des Alarms an. Es kommt manchmal vor, dass durch ein leichtes Erdbeben ein Alarm hervorgerufen wird. Liegt jetzt ein Einbruch vor?“ Diese Situation lässt sich wie in Abbildung 4-4 dargestellt in einem Graphen mit den Knoten „Einbruch“, „Erdbeben“, „Alarm“, „JohnMeldetAlarm“, „MaryMeldetAlarm“ auffassen. Für die Beziehungen zwischen den Knoten werden die folgenden Aussagen herangezogen. „Ein Einbrecher kann den Alarm auslösen“, „Ein Erdbeben kann den Alarm auslösen“, „Der Alarm kann John veranlassen, anzurufen“ und „Der Alarm kann Mary veranlassen, anzurufen“. Dabei werden die Knoten mit den bedingten Wahrscheinlichkeiten für die jeweils zugeordneten Zustände versehen.

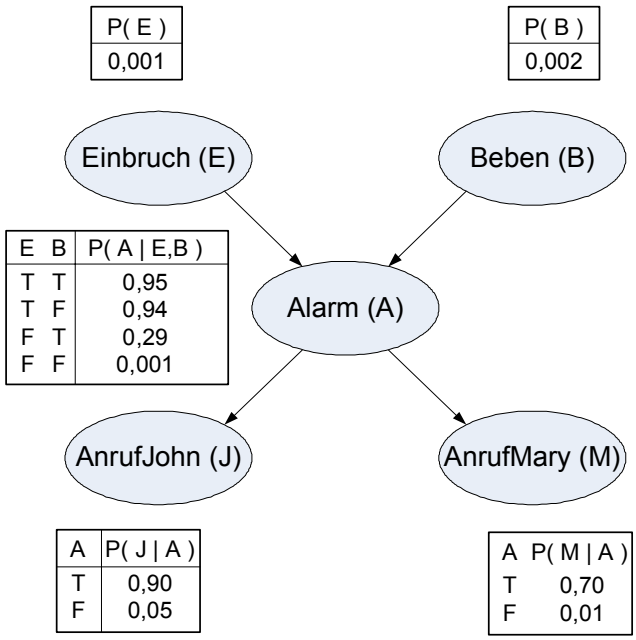


Abbildung 4-4 Bayes-Netz mit verteilten bedingten Wahrscheinlichkeiten

Für die mathematische Betrachtung der Bayes-Netze ist zunächst die totale Wahrscheinlichkeit aus (1) wichtig. Sie besagt, dass die totale Wahrscheinlichkeit eines Ereignisses B der Summe der Erwartungswerte aller seiner durch A_i bedingten Eintrittswahrscheinlichkeiten entspricht:

$$P(B) = \sum_{i=1}^{n} P(B \mid A_i) \cdot P(A_i) \qquad \textbf{(1)}$$

Damit lassen sich über die so genannte Bayes-Formel in (2), die bedingte Wahrscheinlichkeit einer bestimmten Vorbedingung A_k bei Vorliegen von B bestimmen.

$$P(A_k \mid B) = \frac{P(B \mid A_k) \cdot P(A_k)}{P(B)} = \frac{P(B \mid A_k) \cdot P(A_k)}{\sum_{i=1}^{n} P(B \mid A_i) \cdot P(A_i)} \qquad \textbf{(2)}$$

Damit lässt sich bezogen auf die Fahrzeugdiagnose aussagen, welche Komponente A_k des Systems bei Auftreten eines Defektes B mit höchster Wahrscheinlichkeit defekt ist.

Hauptschwierigkeit beim Erstellen der Bayes-Netze sind die Konstruktion der Netze an sich, und die Gewichtung der Kanten mit Wahrscheinlichkeiten. Hier können die Netze auch trainiert werden, wobei sich die Wahrscheinlichkeiten über statistische Analysen von Erfahrungsdaten ermitteln lassen.

4.1.5.3. Neuronale Netzwerke

Neuronale Netze bilden als Modell die Nervenstrukturen des Gehirns von Tieren und Menschen nach: Neuronen sind in der Art eines Netzes miteinander verknüpft. Biologische Neuronen reagieren auf elektrische oder chemische Reize. Neuronen haben üblicherweise mehrere Eingangsverbindungen sowie eine Ausgangsverbindung. Wenn die Summe der Eingangsreize einen gewissen Schwellenwert überschreitet, "feuert" das Neuron. Das bedeutet, dass ein Aktionspotential an seinem Axonhügel ausgelöst und entlang seines Axons weitergeleitet wird. Das ist das Ausgangssignal des Neurons. Eine grafische Darstellung eines künstlichen neuronalen Netzes ist in Abbildung 4-5 dargestellt [Hebb49]. Dabei werden die Eingansneuronen mit Umweltinformationen gereizt und der daraus resultierende Aktivierungszustand der Ausgangsneuronen ist die Antwort des neuronalen Netzes, auf die Stimulation.

Durch die unterschiedliche Gewichtung der Eingangsinformationen an den Neuronen kann das Verhalten des Netzes auf Reize verändert werden. Ausgehend von einer Startverteilung der Gewichtungen wird das Netz trainiert, in dem die Gewichtung verändert werden, es lernt gewissermaßen. Über das Lernen in neuronalen Netzen gibt es verschiedene Ansätze. Die erste neuronale Lernregel wurde 1949 von Donald O. Hebb beschrieben (Hebb'sche Lernregel) [Hebb49].

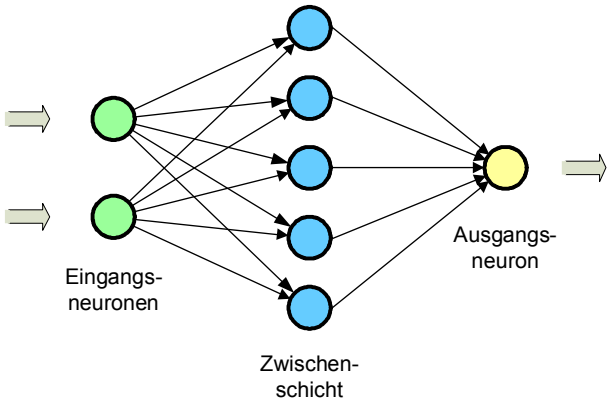


Abbildung 4-5 Vereinfachte Darstellung eines künstlichen neuronalen Netzes

Im Folgenden wird exemplarisch ein wichtiger Lernalgorithmus für neuronale Netzwerke beschrieben, um das Lernen der Netze zu veranschaulichen. Es handelt sich dabei um die so genannte Delta-Regel, die für neuronale Netzwerke ohne Zwischenschicht wie in Abbildung 4-6 dargestellt anwendbar ist [Mach93].

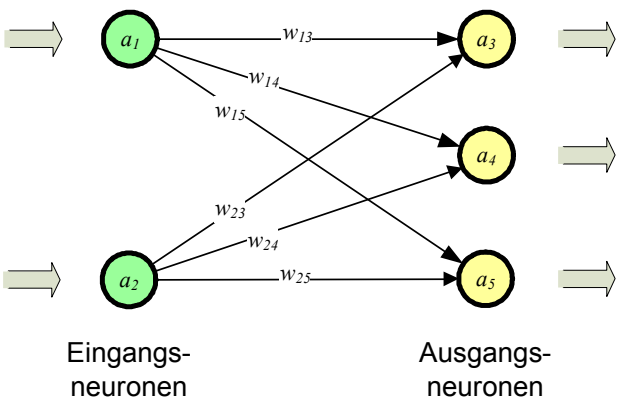


Abbildung 4-6 Neuronales Netz ohne Zwischenschicht

1. Die Eingangsneuronen werden entsprechend der Umweltinformation aktiviert bzw. deaktiviert.

2. Das Netz errechnet die Aktivierung a_i der einzelnen Ausgangsneuronen, indem die Werte aller assoziativen Verbindungen w_{ij} zwischen einem aktiven Eingangsneuron j und einem damit verbundenen Ausgangsneuron i summiert und einer Aktivierungsfunktion f übergeben werden. Die Ausgangsneuronen stehen für das vorherzusagende Ergebnis, hier im Kontext der Automobildiagnose für die einzelnen Fehlerursachen. Die assoziative Verbindung von einem Eingangsneuron zu einem Ausgangsneuron spiegelt das Ausmaß wieder, in dem das Netz einen Zusammenhang zwischen einem bestimmten Symptom und einer Fehlerursache „sieht“. Je stärker die Beziehung zwischen Symptom j und Ursache i, desto höher der Wert von w_{ij}. Die Aktivierungsfunktion f ist je nach Anwendungsfall auszuwählen. Gebräuchliche Funktionen sind z.B. die lineare Aktivierung, die binäre Schwellwertfunktion und die logistische Funktion.

$$a_i = f\left(\sum_j w_{ij} * a_j\right) \quad \textbf{(3)}$$

3. Es wird die Abweichung δ_i bestimmt, inwieweit die errechnete Aktivierung der Ausgangsneuronen a_i mit einer vorgegebenen Zielaktivierung $a_{i,soll}$ übereinstimmt. Die Zielaktivierung stellt die korrekte Netzantwort dar.

$$\delta_i = a_{i,soll} - a_i \quad \textbf{(4)}$$

4. Anpassung der assoziativen Verbindungen: Zu jeder assoziativen Verbindung wird die Differenz zwischen Ziel- und errechneter Aktivierung hinzuaddiert und multipliziert

mit einem Lernfaktor ε, der kleiner als eins ist, z.B. 0.003. Dieser Faktor stellt die Lernrate dar und beeinflusst wesentlich die Lerngeschwindigkeit. Es werden jedoch nur die assoziativen Gewichte zu den aktiven Eingangsneuronen verändert. Der Grund ist, dass nur die aktiven Neuronen zu dem Fehler, der sich in der Differenz zwischen errechneter Aktivierung und Zielaktivierung ausdrückt, beigetragen haben.

$$\Delta w_{ij} = \varepsilon \delta_i a_j \quad \textbf{(5)}$$

5. Vorgeben eines neuen Eingangssignals und Wiederholung des Lernzyklus.

Die besondere Eigenschaft neuronaler Netze besteht darin, dass sie komplexe Muster lernen können, ohne dass eine Abstraktion über die diesen Mustern zugrunde liegenden Regeln stattfindet. Das richtige Trainieren eines neuronalen Netzes ist Voraussetzung für seinen möglichen Lernerfolg.

Ein Vorteil der neuronalen Netze besteht darin, dass vor dem Lernen diese Regeln nicht entwickelt bzw. programmiert werden müssen. Auf der anderen Seite ist dies auch ein Nachteil, da aus einem gut funktionierenden neuronalen Netz nicht die Logik ermittelt werden kann, die seinen Lernerfolg ausmacht. Auch kann der optimale Netzaufbau nur aufgrund von Erfahrung und empirisch gefunden werden. Es ist offensichtlich, dass Netze mit sehr geringer Anzahl von Neuronen der geforderten Komplexität der Aufgabe nicht gewachsen sind. Trotzdem verschlechtert eine zu große Zahl von Neuronen das Ergebnis, weil Trainingsdaten hier nur gespeichert werden, während die dahinter liegende „Regel“ vom Netz nicht mehr erfasst wird. Lernalgorithmus, Auswahl der Trainingsdaten und die Anzahl von Trainingsdurchläufen verändern die Güte der Resultate stark. Gleichermaßen nachteilig ist somit, dass sich das „richtige“ Verhalten neuronaler Netze nicht verifizieren lässt, es sei denn, man stimuliert es mit allen möglichen Eingangsmustern und vergleicht es mit der zugehörigen gewünschten Reaktion

Für allgemeine Systemdiagnosen im hochkomplex vernetzten Automobil werden künstliche neuronale Netze damit bisher nicht eingesetzt. In begrenzten Teilbereichen wie der Katalysatorüberwachung, der Motordiagnose oder der Sensorüberwachung jedoch können die Stärken der neuronalen Netze im Bereich der Mustererkennung zum Tragen kommen [Ayeb00].

4.1.6. Nicht näher einzuordnende Diagnoseverfahren

Nicht alle Diagnoseverfahren lassen sich unter anerkannten Begriffen zusammenfassen, dazu zählen vor allem Verfahren, die erst in jüngerer Vergangenheit publiziert wurden. Die folgenden Diagnoseverfahren beschreiben solche Ansätze.

4.1.6.1. Systemdiagnoseverfahren

In der Offenlegungsschrift DE10051781A1 [Bäke00] schlagen die Erfinder ein zweistufiges Systemdiagnoseverfahren vor. Zunächst werden Einzeldiagnosen für Komponenten bzw. Funktionen erstellt. In einem zweiten Schritt werden diese zu Systemdiagnosen verknüpft (s. Abbildung 4-7.

Als besonderes Merkmal wird angeführt, dass die Einzeldiagnosen unabhängig von der Systemdiagnose sind. Damit sollen die vielen möglichen Ausstattungsvarianten eines Fahrzeugs auch von der Diagnose unterstützt werden, ohne dass die Systemdiagnose explizit darauf abgestimmt sein muss.

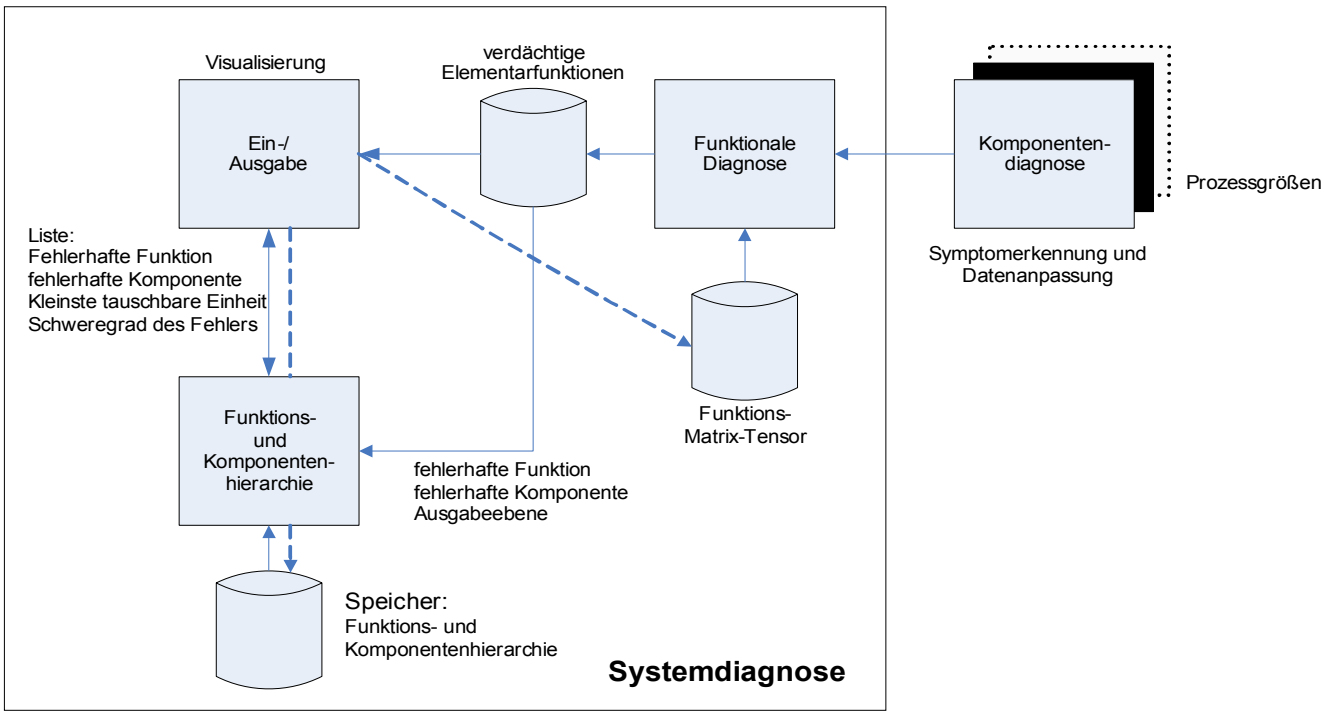


Abbildung 4-7 Systemdiagnose auf Basis von Komponentendiagnosen [Bäke00]

Als Vorteil erweist sich bei diesem Verfahren auch, dass keine einheitlichen Berechnungsvorschriften für die Komponentendiagnosen vorgegeben sind. Es können daher für jede Komponente das am Besten passende Verfahren eingesetzt werden, sei es Fuzzy-Logik, modell- oder wissensbasierte Diagnose etc. Wichtig ist lediglich die Abbildung des Zustands einer

Komponente auf die Werte „Defekt“, „Nicht Defekt“ oder „Unbekannt“. Damit wird auch das Problem der breitbandigen Übermittlung von Daten zur zentralen Systemdiagnose gelöst, wenn ein hochdynamisches Modellverhalten vorliegt. Die dynamischen Vorgänge werden von der Komponentendiagnose direkt vor Ort bearbeitet und in die Funktionszustände überführt.

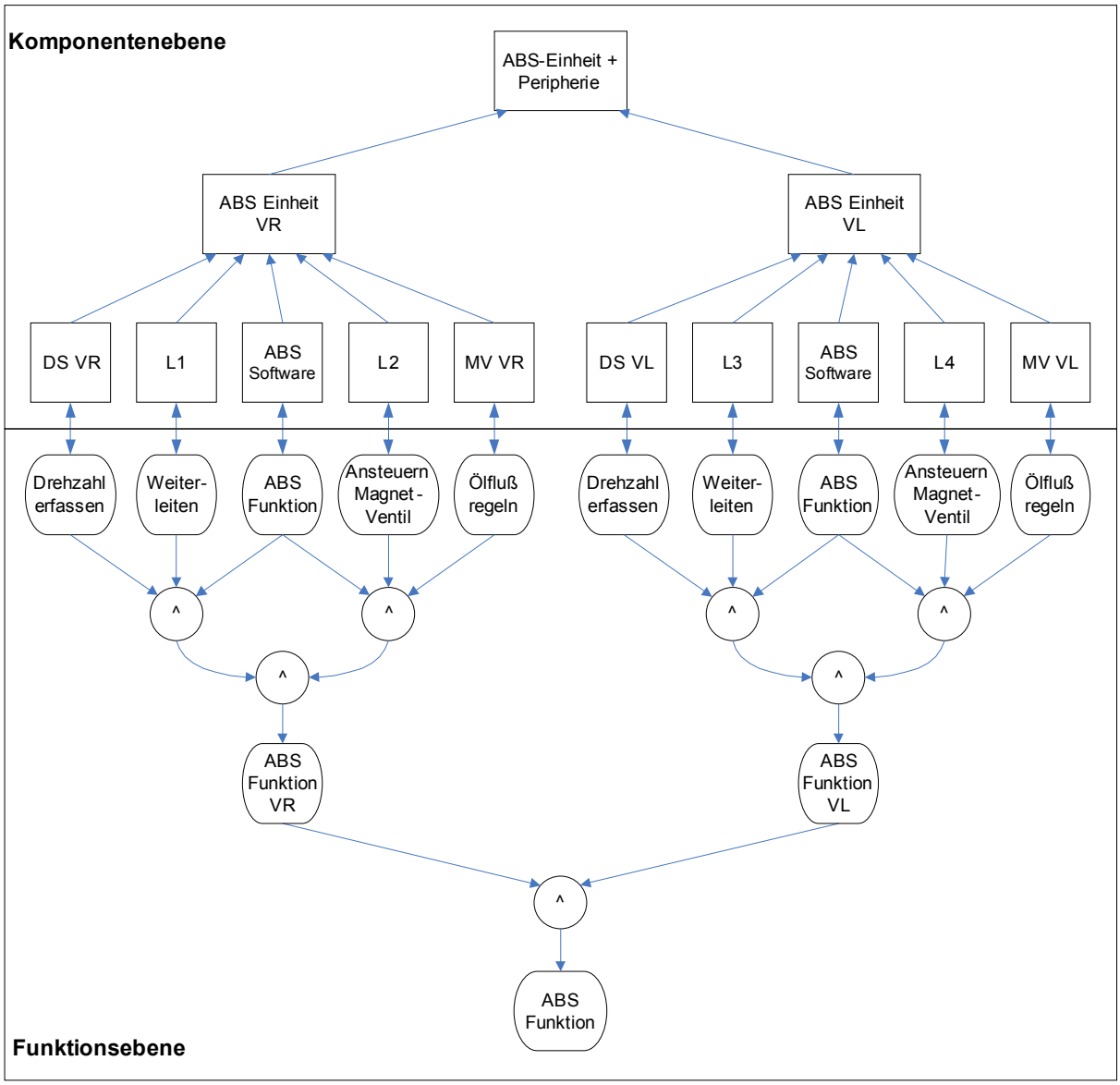


Abbildung 4-8 Komponenten- und Hierarchieebenen [Bäke00]

In dem so genannten Funktions-Matrix-Tensor sind die Beziehungen zwischen Komponentenzuständen bzw. Funktionszuständen zu Systemgrößen im Speicher des Systemdiagnoseprogramms abgelegt. Anhand dieser Matrix können fehlerhafte und fehlerverdächtige Elementarfunktionen bestimmt werden. Zusammen mit der zusätzlich hinterlegten Funktions-

und Komponentenhierarchie lassen sich damit der Schweregrad des Fehlers sowie die kleinste tauschbare Einheit ermitteln (s. Abbildung 4-8).

4.1.7. Gewinnung von Diagnosewissen

Es gibt, wie die obige (unvollständige) Auswahl an Diagnoseverfahren zeigt, etliche unterschiedliche Ansätze zum Diagnoseverfahren. Es gibt aber auch viele Autoren, die sich mit dem Wissenserwerb für bestehende Diagnosesysteme befassen. In verschiedenen Publikationen werden drei generell unterschiedliche Wege beschritten, um Daten für die Diagnose zu generieren [Ring99]. In frühen Untersuchungen versuchte man indirekt Diagnosewissen zu akkumulieren. Dazu befragten die Diagnoseautoren in Interviews die Fachspezialisten und gaben daraufhin die Antworten formalisiert in das Diagnosewerkzeug ein [Benn82][Curt91][Kahn88].

Später wurde das Ziel verfolgt, durch sehr verständliche und komfortable Eingabemasken den Entwicklungsexperten selbst zu befähigen, die notwendigen Daten in die Diagnosesysteme direkt einzuarbeiten [Marb91].

Auch sollte die Wissensakquisition die diagnostisch relevanten Informationen z.B. aus Schaltplanerstellungssystemen automatisch in Diagnosesysteme übernehmen. Dies führte zu dem Problem, dass das elektrische System eines Automobils nicht nur aus passiven elektrischen Schaltungen bestehen, sondern auch aktive Komponenten z.B. in Form von Steuergeräten mit darauf ablaufender Software vorhanden sind [Miss96][Frit96].

Durch die zunehmende Vernetzung der Werkstätten können die Fahrzeughersteller Feldprobleme schneller identifizieren und in die Diagnosesysteme vermehrt statistische Aussagen einfließen lassen. [John03]

Durch die im Entwicklungsprozess frühere und engere Kopplung der Funktionsentwicklung mit der Diagnosefunktionsentwicklung und der gleichzeitigen weitergehenden Formalisierung von Anforderungen, kann die Diagnose unter geringerem Zeitdruck mit verlässlichen Informationen erstellt werden. [Rett05]

4.1.8. Spezielle Aspekte der Diagnose in der Automobilindustrie

Im Folgenden sollen die besonderen Eigenschaften dargestellt werden, die die Automobilindustrie von der Diagnose fordert, und welchen zusätzlichen Umfang zur eigentlichen Fehlerfindung und Behebung der Begriff „Automobil-Diagnose“ umfasst.

Wichtige Arbeiten im Automotive-Bereich werden seit 1994 auf dem Gebiet der Standardisierung der Datenverwaltung und Diagnosekommunikation mit der Association for Standardisation of Automation and Measurement (ASAM) in einem herstellerübergreifenden Konsortium durchgeführt Unter anderem sind für die Diagnose die Anbindung der Steuergeräte über das im ASAM Standard MCD (Measurement, Calibrations, Diagnostics) beschriebene Verfahren besonders relevant und immer weiter verbreitet. Hier geht es darum, die Schnittstellen für den Steuergerätezugriff bei Messungen, Applikationsaufgaben und für Diagnosezugriffe zu definieren und international zu vereinheitlichen [Stef04][ASAM06].

4.1.8.1. Zugriff auf Steuergerätedaten

Wesentlicher Teil der Diagnose, wenn auch nicht der eigentlichen Fehlersuche, ist der Kommunikationszugang zum Fahrzeug und dessen relevanten Daten. Dieser Zugang wird von den Fahrzeugherstellern immer weiter standardisiert, schon aufgrund der gesetzlichen Forderung, auch freien Werkstätten uneingeschränkten Zugang zum elektronischen Innenleben eines Automobils zu gewähren.

In den meisten Fahrzeugen wird der Zugang zu den Steuergeräten zumindest über die so genannte K-Leitung ermöglicht. Es handelt sich dabei um einen seriell arbeitenden Datenbus, der bei einer dynamisch festgelegten Datenrate Diagnose-Kommunikation ermöglicht. Die entsprechenden Kommunikationsdienste werden in der ISO 9141-2 definiert, erlauben aber große herstellerspezifische Unterschiede [ISO9141].

Der recht frei interpretierbare Standard ISO 9141-2 wurde daher abgelöst durch das Keyword Protocol 2000 (KWP 2000), definiert in der ISO 14230/Road Vehicles — Diagnostic Systems [ISO14230]. Für die herstellerspezifische Diagnose wird üblicherweise eine Teilmenge der in der ISO 14230 definierten Diagnostic Services implementiert. Mit ISO 14230 Keyword Protocol 2000 und ISO 15765 Diagnostics on CAN ist die Diagnoseschnittstelle der Steuergeräte nun weitgehend standardisiert [ISO15765].

Dennoch ist es aufgrund der dem Hersteller möglichen eigenen Interpretation der Standards noch nicht soweit, dass sich mit einem bestimmten Testgerät alle möglichen Fahrzeuge aller Hersteller ausgelesen werden können. Es gibt immer ein wenig unterschiedliche Steckertypen und –belegungen sowie unterschiedliche Interpretationen der Protokolldaten.

Gelingt jedoch der kommunikative Zugang zum Automobil, so sind die dort zu findenden Fehlerspeichereinträge (Diagnostic Trouble Code, DTC) bezüglich der Motordaten im Rahmen der OBD herstellerübergreifend standardisiert.

Aufgrund des Bedarfs der Fahrzeughersteller, auch im Verlauf des Fahrzeuglebens neue Software in das Fahrzeug zu bringen, z.B. um Softwarefehler zu beheben, wurde zusätzlich zur K-Leitung durch einen CAN-Zugang zum Fahrzeug mit 500kBit/s der Zugriff auf das Fahrzeugnetzwerk stark beschleunigt. Dieses so genannte Flashen ist mittlerweile integraler Bestandteil des Diagnosebegriffs im Automobilbereich. In der Zukunft soll der Datenzugang immer variabler gestaltet werden, so dass auch ein Fernzugriff auf das Automobil ermöglicht werden könnte.

Teil der Aufgaben der Diagnosekommunikation ist bei allen Fahrzeugherstellern die Variantenkodierung der Steuergeräte. Dies geschieht vornehmlich bei der Produktion am Bandende, kann aber über die Testergeräte auch in den Werkstätten vorgenommen werden. Dabei können beispielsweise ein Bestätigungssignal der Hupe beim Betätigen der Zentralverriegelung zusätzlich zum Bestätigungsleuchten der Blinker parametriert werden.

Die korrekte Konfiguration des Testers vorausgesetzt, wird auch eine Stimulierung des Systems über die Diagnosekommunikation ermöglicht. Beim so genannten Stellgliedtest können die Aktoren der Steuergeräte einzeln dauerhaft angesteuert werden, um ohne aufwändigen Teileausbau Fehler in den Treibern, Leitungen und den Bauteilen leicht identifizieren zu können.

Über die standardisierte Datendefinition anhand der ASAM Vorgaben wird ein sehr flexibler Zugang zu den Steuergeräten ermöglicht [ASAM06]. Dabei ist vor allem die enge Integration der Diagnosedaten schon während der Entwicklung ein wesentlicher Aspekt. So können die Entwickler während der Steuergeräteentwicklung interne Variablen als von außen zugänglich markieren. Damit werden die entsprechenden Datenzuordnung in Form von Variablennamen und -adresse im Steuergerät sowie das verwendete Datenformat und die physikalische Einheit in eine Datenbank eingetragen, und für den Tester verwendbar. Für diese Art der Diagnosedatenhaltung gibt es z.B. mit dem Produkt Candela von Vector Informatik umfangreiche Programme, die eine sehr komfortable Verwaltung der ASAM-MCD-Daten eines Herstellers bieten. Ebenso könne mit einem solchen System die von den Steuergeräten generierten Fehlercodes effizient verwaltet werden [Vect05].

4.1.8.2. Entwicklungs- und Herstellungskosten für Diagnosefunktionen

Ein Aspekt für die Entwicklung von Diagnosefunktionen ist es, Fehlfunktionen frühzeitig erkennen zu können und diese kostengünstig und schnell zu beheben. Aufgrund weit reichender Gewährleistungsansprüche der Kunden und der Erhaltung der Kundenzufriedenheit hat

hier auch der Automobilhersteller ein wirtschaftliches Interesse daran, zielführende Diagnosefunktionen zu realisieren. Dennoch ist hier bei der technischen Auslegung des Systems abzuwägen, welche Diagnosetiefe erreicht werden soll, denn eine größere Diagnosetiefe ist meist mit höheren Entwicklungs- und Stückkosten verbunden. Wichtig ist daher vor allem, mit angemessenem Aufwand Diagnosen derart zu finden, dass eine tauschbare Komponente möglichst schnell und eindeutig (s. Kosten im Kundendienst) identifiziert werden kann. Einmalige hohe Entwicklungskosten für die Implementierung eines Werkzeugs, das präzise Diagnosen erzeugen kann, sind aufgrund der hohen Produktionsstückzahlen von Automobilen schnell akzeptabel, wenn die damit erreichte Genauigkeit der Diagnose im Kundendienst und in der Produktion verbessert werden kann. Es ist auch möglich, die Diagnose an Bord des Fahrzeugs bewusst einzuschränken und auf einige wenige Funktionen zu beschränken, damit die ressourcenhungrige übergeordnete Systemdiagnose auf leistungsfähigen PCs nur mit niedrigen Produktionsstückzahlen für den Kundendienst durchgeführt werden kann.

4.2. Abwägung der Modellierungsansätze gegeneinander

Eine Diagnoselösung für das Kraftfahrzeug hat spezifische Anforderungen, die in den vorigen Abschnitten näher erläutert wurden. Im Folgenden werden die zur Verfügung stehenden Diagnoseverfahren für Ihren Einsatz im automobilen Entwicklungsprozess und im Kundendienst bewertet und miteinander verglichen. Vorrangige Entscheidungskriterien sind dabei die möglichst kostengünstige Anwendung in der Praxis und die effiziente Entwicklung der Diagnoseregeln bzw. –modelle.

Prozedurale Diagnose ist in den Werkstätten die derzeit am häufigsten eingesetzte Lösung für die Diagnose. Allerdings ist sie aufgrund ihres Entwicklungsaufwandes den immer umfangreicheren elektronischen Systemen eines Automobils nicht gewachsen. Auch wenn zur Zeit noch viele Diagnosesysteme auf von Expertenhand generierten Fehlerbäumen basieren, ist doch das Potential dieser Diagnosesysteme ausgereizt. Die mangelnde Flexibilität der Fehlerbäume gegenüber unterschiedlichen Ausstattungsvarianten und Modellreihen ist ebenfalls ein wesentlicher Punkt, der gegen den Einsatz der prozeduralen Diagnose spricht. Die prozedurale Diagnose wird dennoch in absehbarer Zeit trotzdem ein wichtiger Bestandteil der Automobildiagnose bleiben. Es ist aber möglich, dass Teile durch generisch erzeugte Datenbasen erzeugt werden, während andere Teile von Experten aus der Werkstatt eingegebenes Erfahrungswissen angereichert werden.

Ähnlich verhält es sich bei der *regelbasierten* Diagnose. Die Erstellung der notwendigen Regeln kann von den Experten bei den heutigen komplexen Systemen nicht mit ausreichender Systemabdeckung und Genauigkeit realisiert werden, denn der Erstellungsaufwand bei steigender Komplexität des Systems ist immens. Allerdings kann dieser Ansatz bei automatisierter Erstellung von Diagnoseregeln (z.B. aus der modellbasierten Diagnose heraus) eine schnell rechnende und ressourcensparende Diagnoselösung auch auf einfachen Steuergeräten darstellen.

Neuronale Netzwerke werden in der Diagnose eingesetzt, vor allem wenn Muster in Messdaten erkannt werden sollen. Dies rechtfertigt ihren Einsatz in Systemen zur Abgasüberwachung, jedoch ist ein Einsatz, der das umfassende elektrische System eines Automobils mit seiner Vielzahl unabhängiger Messgrößen und unterschiedlichsten Fehlermöglichkeiten überwacht derzeit nicht vorstellbar. Ein weiterer wesentlicher Kritikpunkt an den neuronalen Netzen ist die prinzipiell bedingt fehlende Kausalkette zur Diagnose.

Die *modellbasierte* Diagnose ermöglicht einen detailreichen Modellaufbau, der dem individuellen elektrischen System eines Automobils entspricht. Die Möglichkeit, modellbasierte Diagnose in STEP-X einzusetzen, bietet einige auf der Hand liegende Vorteile, die den probeweisen Einsatz im Rahmen der Arbeiten von STEP-X rechtfertigten. Zum Einen sollten damit Synergien aus dem ohnehin modellbasierten Entwicklungsansatz von STEP-X genutzt werden, zum Anderen verspricht der Einsatz modellbasierter Diagnose auch die Auffindung bisher unbekannter Fehler im System. Positiv fällt auch ins Gewicht, dass ein minimales Systemmodell lediglich das Sollverhalten aller beteiligten Komponenten enthält. Damit lässt sich ein System sehr anschaulich beschreiben, ohne dass alle möglichen Fehlerszenarien überlegt sein müssen. In der Umsetzung stellte sich allerdings heraus, dass die Systemmodelle sehr umfangreich und detailliert zu formulieren waren. So müssen Steuergeräteinterna sowie Steuergeräteumwelt mit sehr konkreten Modellen ihres Verhaltens modelliert werden. Bei diesem versuchsweisen Einsatz der modellbasierten Diagnose wurde davon ausgegangen, dass in einem modellbasierten Entwicklungsprozess die Modelle für eine modellbasierte Diagnose mit nur geringem Mehraufwand weiter genutzt werden können. Diese Sichtweise verschließt sich aber der Tatsache, dass die Steuerungsmodelle der Steuergeräte fundamental unterschiedlich zu den Modellen der modellbasierten Diagnose sind. Während in den Steuerungsmodellen das logische Verhalten des Steuergerätes beschrieben wird, wird für die modellbasierte Diagnose zusätzlich vor allem ein Modell der Außenwelt eines Steuergerätes benötigt. So interessieren hier für ein Fehlverhalten eines Motors z.B. seine Zuleitungen und die verwendeten elektri-

schen Treiberstufen. Diese werden für die Erstellung der Software gänzlich vernachlässigt, hier werden lediglich die Portpins zur Ansteuerung der Motorelektronik bedient, und davon ausgegangen, dass sich daraufhin der Motor dreht.

Im Rahmen der exemplarischen Umsetzung stellte sich weiterhin heraus, dass die vorhergesehenen Fehlermodi der Modellkomponenten zwar für zutreffende Diagnosen genutzt werden konnten, aber der Aufwand, das Systemmodell zu erstellen war unverhältnismäßig hoch und rechtfertigte daher keinen umfassenden Einsatz für das Gesamtsystem Kraftfahrzeug. Zusätzlich stellte sich heraus, dass durch den Einsatz der modellbasierten Diagnose zwar bei der Modellerstellung genaue quantitative Modelle verwendet wurden, aber auf dem Steuergerät lediglich davon abgeleitete einfache qualitative Entscheidungsbäume hervorgingen, die ähnlich wie beim manuellen Erstellen von Diagnosefunktionen stückweise für das tatsächliche System optimiert wurden. Auch der festgestellte hohe Rechenaufwand und der damit verbundene Zeitaufwand bei jeder Erstellung des qualitativen Modells aus dem quantitativen Modell stellte sich als Hindernis für einen praktischen Einsatz heraus.

Aufgrund der beschränkten Anzahl von Messwerten, die einem Steuergerät zur Verfügung steht, konnte auch die Diagnosequalität gegenüber manuell implementierten Diagnosefunktionen auf dem Steuergerät nicht deutlich verbessert werden.

Im Vergleich zur herkömmlichen Diagnose wurde dabei festgestellt, dass ein Einsatz modellbasierter Diagnose wesentlichen Mehraufwand bei der Entwicklung bedeutet. Insbesondere die Detailtiefe, die für die Erstellung der Modelle gefordert wurde, war ungleich höher als es der Gewinn an Diagnosegenauigkeit rechtfertigte. Damit kommt ein Einsatz zumindest im Rahmen einer Systemdiagnose vorerst nicht in Betracht, und wurde deshalb im Projekt auch nicht weiter verfolgt.

Die bekannten *probabilistischen Netzwerke* bieten einen aussichtsreichen Ansatz, Diagnosesysteme für das Automobil zu erstellen. Durch die Betrachtung von Strukturen, die das Automobil selbst vorgibt und die im Entwicklungsprozess ohnehin erstellt werden, bietet dieses Verfahren Möglichkeiten, um kostengünstige und für ein individuelles Fahrzeug parametrierte Diagnosen zu erhalten. Ein Nachteil ist jedoch die Notwendigkeit, Ausfallwahrscheinlichkeiten für einzelne Strukturelemente bestimmen zu müssen. Auch wenn ein Einsatz unter der sicheren Einschätzung eines konstanten Einflusses machbar schiene, bedeutet doch die zusätzliche Restriktion, rückwirkungsfreie Systeme zu betrachten eine Forderung, die heutige vernetzte Systeme nicht erfüllen können. Dennoch zeigen aktuelle Forschungen bei zwei großen

Fahrzeugherstellern [Borg04][Luek04], dass in dieser Methode ein viel versprechendes Potential liegt.

Eigenschaft	**Prozedurale und regelbasierte Diagnose**	**Probabilistische Netzwerke**	**Modellbasierte Diagnose**	**Neuronale Netzwerke**	**Strukturanalyse**
Verarbeitungszeit der Diagnosefunktionen	Schnell	Schnell	Je nach Systemumfang	Schnell	Schnell
Nachvollziehbarkeit der Ergebnisse und Zwischenschritte	Wenn vom Entwickler vorgesehen	Anhand der Systemstrukturen	Teil des Analyseergebnisses	Nicht möglich	Anhand der Systemstrukturen
Abdeckungsgrad des diagnostizierten Systems	Bei komplexen Systemen gering	Je nach Auflösung der zugrunde liegenden Systembeschreibung	Je nach Aufwand	Bei komplexen Systemen gering	Je nach Auflösung der zugrunde liegenden Systembeschreibung
Kenntnis der Modellparameter des diagnostizierten Systems	Mittel	Mittel	Sehr hoch	Keine	Mittel
Flexibilität des Systems	Gering	Hoch	Hoch	Gering	Hoch
Exaktheit der Ergebnisse	Entwicklungsabhängig, bei komplexen Systemen gering	Mittel	Hoch	Je nach System	Gering
Entwicklungsaufwand für die Diagnose	Sehr hoch	Hoch	Sehr hoch	Hoch	Gering
Automatisierbarkeit	Keine	Teilweise	Keine	Mittel	Hoch
Universelle Einsetzbarkeit	Ja	Ja	Ja	Ja	Ja
Hardwareaufwand	Je nach Vorgabe	Je nach Vorgabe	Je nach Vorgabe	Je nach Vorgabe	Je nach Vorgabe
Prozessdatenumfang Onboard	Gering	Gering	Gering	Hoch	Gering

Tabelle 4-1 Klassifizierung von Diagnosesystemen

Ähnlich wie bei den probabilistischen Netzwerken wird bei der *Strukturanalyse* mit Komponentenstrukturen des Systems gearbeitet. Dieser Ansatz ist zwar in der Literatur bekannt, ihm wurde aber bisher keine Bedeutung beigemessen, was vor allem an den dort beschriebenen begrenzten Umfängen der Strukturanalysen liegt. In [Lemm94] und [Bäke00] werden diese

Ansätze zwar erläutert, aber nicht konsequent genug verfolgt. So wurden in den bisherigen Arbeiten lediglich Teilaspekte wie das elektrische System eines Fahrzeugs betrachtet.

Lediglich die Strukturanalyse bietet die Möglichkeit, mittels rasch durchführbarer Analysen auch bei variantenreichen Fahrzeugtypen spezifisch und automatisiert zu funktionieren, ohne das Gesamtsystem Fahrzeug zu vernachlässigen. Eine solche Flexibilität bei guten Diagnoseergebnissen bietet keines der anderen betrachteten Verfahren, weshalb die Strukturanalyse als Basis für die vorgestellten Arbeiten heran gezogen wird.

Zusammenfassend stellt Tabelle 4-1 die beschriebenen wesentlichen Eigenschaften der einzelnen Verfahren in Kurzform dar.

5. Realisierter Diagnoseansatz

Basierend auf den vorhergehenden Abschnitten soll im Folgenden dargestellt werden, wie der im Rahmen der Arbeiten entstandene Diagnoseansatz realisiert wurde, und aus welchen Gründen gerade dieses Vorgehen gegenüber anderen Ansätzen besseren Erfolg zu versprechen scheint.

Bereits in der Einführung wurde dargelegt, dass die heutigen elektronischen Systeme im Automobil zu komplex werden, als dass die zugehörigen Wissensbasen für Diagnosesysteme allein von Experten erstellt werden könnten. Daher kommen prinzipiell nur Diagnoseverfahren in Frage, die Entwicklungsinformationen automatisch weiter nutzen, Fehler bzw. Fehlerauswirkungen vorhersagen und dabei unterschiedliche Fehlermodi der Komponenten integrieren.

Diagnosesysteme werden im Allgemeinen zweigeteilt erstellt. Zum Einen wird eine Datenbasis erstellt, auf dem das Diagnosesystem arbeitet. Diese Aufgabe erfolgt im Rahmen der vorgestellten Arbeiten weitgehend automatisch. Zum Anderen muss die Laufzeitkomponente des Diagnosesystems mit dem Diagnosealgorithmus und der grafischen Benutzeroberfläche implementiert werden. Diese beiden Aufgaben wurden im Rahmen einer ganzheitlichen Betrachtung in der vorliegenden Arbeit exemplarisch durchgeführt und beschrieben.

Um die behandelten Aufgabengebiete dieser Arbeit technisch nachvollziehbar und generell zu gestalten, beschränken sich die Untersuchungen vor allem auf die automatisiert ablaufende Wissensbeschaffung aus CAE-Daten, um eine Grundlage für ein Diagnosesystem zu schaffen. Die Ordnung der Kandidatenlisten hinsichtlich konkreter Ausfallwahrscheinlichkeiten von Komponenten kombiniert mit Tauschkosten muss bei der Erweiterung der Methoden für den praktischen Einsatz berücksichtigt werden.

In diesem Abschnitt wird daher folgendes Vorgehen bezüglich des Aufbaus und der Anwendung von Diagnosewissen vorgestellt:

- Automatisierte Übernahme der relevanten Daten aus den Entwicklungssystemen
- Erstellung einer zugehörigen Wissensbasis für die Diagnose

  Diese Wissensbasis soll in einem simulierten Ablaufsystem einer Werkstatt genutzt werden. Dazu sind Module zur Kommunikation mit dem Fahrzeug, die Benutzer-Schnittstelle sowie Protokollierungsmechanismen in das Diagnosesystem zu integrieren. Die eigentli-

che Inferenzmaschine nimmt damit nicht einmal den größten Teil des Diagnosesystems ein.

Im Mittelpunkt der Betrachtungen bei der Erstellung der Wissensbasis und der Nutzung des Diagnosesystems stehen folgende Aspekte:

- Vollständige Abbildung des Fahrzeugs aus diagnostischer Sicht.
- Wahrscheinlichkeitsbasierte priorisierte Ausgabe von Fehlerkandidaten.

5.1. Methodisches Konzept

In diesem Abschnitt soll gezeigt werden, in welcher Relation die bisher bekannten Diagnoseverfahren zu den Darstellungen in dieser Arbeit stehen. Dabei soll vor allem gezeigt werden, dass nicht allein das Diagnoseverfahren, sondern vor allem auch seine frühe Integration in den Entwicklungsprozess und die mehrfache Nutzung von Entwicklungsdaten für unterschiedliche Entwicklungsaspekte wesentliche Stützpfeiler für moderne Diagnosesysteme darstellen.

Vorrangiges Ziel dieser Arbeit ist die Einbeziehung der Diagnose in einen strukturierten Entwicklungsprozess. Am Beispiel des Projektes STEP-X soll diese Integration gezeigt werden. Durch Vorgabe von Regeln und Schemata für die Entwickler wird die Erstellung aller Aspekte der Diagnose, d.h. von der Entwicklung bis zum Kundendienst, gut dokumentiert, einfach zu implementieren, jederzeit nachprüfbar und in der Praxis zuverlässig. Durch die Vorgabe von Implementierungsrichtlinien werden Entwickler dazu gezwungen, sich an bewährte Designregeln zu halten, und die Diagnosefunktionalität schon früh im Entwicklungsprozess zu berücksichtigen.

Die Einbettung in den Entwicklungsprozess ist zusammen mit der Definition eines geeigneten Diagnosealgorithmus der wesentliche Schwerpunkt dieser Arbeit. Oft wurden in der Vergangenheit diese Themen getrennt behandelt, was im Hinblick auf die Praxistauglichkeit nicht optimal ist.

Um dieses Ziel zu erreichen wurden unterschiedliche Ansätze aufgegriffen und weiter ausgebaut. Hinsichtlich des zugrunde liegenden Diagnosewissens wird erweiternd zu bestehenden Systemen sowohl die elektrische Verschaltung der Komponenten untereinander, als auch die Steuergerätesoftware und die Kommunikation in die Überlegungen einbezogen und automatisiert ausgewertet. Anhand eines Demonstrations-Diagnosesystems werden die implementierten Verfahren getestet und bewertet.

Basis aller verwendeten Diagnosemodelle sind die Entwicklungsmodelle des in STEP-X entwickelten Komfortsystems.

Das Auswerteprogramm basiert auf der Extraktion von gespeicherten Daten aus den zugehörigen Entwicklungsprogrammen. Im Einzelnen sind dies zunächst Daten aus mdl-Dateien für die Matlab/Simulink-Modelle. Für die Kommunikationsbeschreibungsdateien wird das .dbc-Dateiformat der Firma Vector-Informatik eingelesen und verarbeitet. Hardwarebeziehungen können sowohl als Teil der Spezifikation aus einem DOORS-Repository heraus gelesen, als auch anhand von Netzlisten in das Analysewerkzeug importiert werden.

Das Analyseprogramm selbst ist in reinem Ansi-C realisiert und ablauffähig auf allen gängigen Systemen mit Microsoft Windows 2000 Betriebssystem. Die Kompilierung für andere Systeme ist aufgrund der Beschränkung auf Ansi-C ohne weitere Einschränkungen möglich. Aufgrund der statischen Analyse der Entwicklungsmodelle ohne Simulationsbedarf, ist das Analyseprogramm nicht auf ein ablauffähiges Entwicklungssystem angewiesen. D.h. der Anwender des Diagnoseverfahrens benötigt keine Lizenz für weitere Entwicklungswerkzeuge wie Matlab/Simulink, CANoe o.ä.

In einem ersten Ansatz von STEP-X wurden alle Diagnosen in einem Gesamtmodell zentral in einem Steuergerät berechnet. Die Modellerstellung und der Ressourcenbedarf für ein solches Vorgehen stellte sich dabei als unverhältnismäßig aufwändig heraus. Daher werden in dieser Arbeit zur Einsparung von Rechenzeit und Kommunikationsbandbreite die Diagnosen für das elektrische Teilsystem eines Steuergerätes dezentral auf den Steuergeräten ermittelt, und lediglich übergeordnete Systemdiagnosen zentral berechnet.

Um die Anforderung nach Dezentralisierung zu berücksichtigen, ist es notwendig, einen Teil der Diagnoseaufgaben auf den Funktionsentwickler eines Steuergerätes zu übertragen. Dieser soll z.B. für die Hardwareschnittstellen des Steuergerätes einfache Überwachungsfunktionen entwickeln, wie das bisher anhand von Fehlercodes realisiert wird. Damit werden Sensoren und Aktoren von der Anwendung ausgewertet und auf korrekte Funktionalität geprüft. Da diese Funktionalität ohne umfassende zusätzliche Diagnosemodellbildung auskommt, ist es so möglich, auf einem einfachen Weg zu Einzeldiagnosen für Teilfunktionen zu kommen. Da sich die Fehlererkennung auf dem Steuergerät auf einfache mathematische Funktionen wie Grenzwertverletzungen etc. beschränkt führt, dies nur zu geringem Ressourcenbedarf im Steuergerät.

Die Abgrenzung, welche Teile des Modells von dem Entwickler autark für die Diagnose bearbeitet werden können, wird anhand der Komplexität des betroffenen Teilsystems getroffen.

Nur klar abgegrenzte und einfache Funktionen sollen vom Entwickler mit Diagnoselogik ausgestattet werden.

Daher wurde ein Algorithmus implementiert, der die Bestimmung der Komplexität von Strukturen vornimmt. Damit kann die Unabhängigkeit von Teilstrukturen z. B. von Kommunikationsarchitektur, Bordnetz oder Funktionen, gezeigt werden. Dies soll dem Entwickler helfen, Diagnosen einer Teilfunktion korrekt zu implementieren, da nur unabhängige Systeme vollständig überschaubar sind. Damit wird an den Entwickler des Steuergerätes die Anforderung gestellt, möglichst unabhängige Systemteile zu erstellen, um schwer diagnostisch zu erfassende Fehlerfortpflanzung bzw. Korrelation von Signalen zu vermeiden. So wird es möglich, ohne komplexes Diagnosemodell des Gesamtsystems, eine vollständig überschaubare Teilfunktionalität mit Fehlererkennung auszustatten.

Für das Diagnosesystem soll auch die implementierte Softwarestruktur mit einbezogen werden, denn mit dem Wissen über die Softwarestruktur ist es möglich, die Auswirkungsmöglichkeiten von Fehlern auf die Funktionalität anderer Komponenten zu bestimmen. Solche Pfade der Fehlerausbreitung werden auch als Wirkketten bezeichnet. Diese Information ist z.B. nützlich, wenn es während des Betriebs des Automobils eine nicht ausreichend diskriminierbare Menge von Fehlerkandidaten gibt. In diesem Fall kann die Werkstatt von einem System angeleitet werden, andere Funktionen zu überprüfen, die Fehlerkandidaten entlasten können. Sind die Ergebnisse dieser Überprüfungsfunktion eindeutig konsistent und fehlerfrei, können diese Komponenten als fehlerlos aus der Kandidatenliste gestrichen werden.

Hinsichtlich der Extraktion von kundendienstrelevanter Information aus den Diagnosen ist es wichtig, die Granularität der Komponentenmodelle zu optimieren. So muss einerseits die Rechenzeit für die Berechnung der Fehlerkandidaten auf dem Steuergerät minimiert werden. Andererseits muss die Diagnose dennoch so exakt sein, dass ein für den Kundendienst befriedigendes Ergebnis erzielt wird, damit dieser nur das fehlerhafte Bauteil findet und ggf. austauscht.

5.1.1. Strukturelle Diagnose

Die Abwägung der unterschiedlichen Diagnoseverfahren gegeneinander führte zu der Erkenntnis, dass es bisher keine ausreichende prozesssichere Systemdiagnoselösung für den automotiven Einsatz gibt. Aus dieser Erkenntnis heraus wird in dieser Arbeit ein neuer kom-

binierter Ansatz verfolgt, um die Stärken einzelner Diagnoseverfahren zu verwenden, und dabei möglichst ihre Schwächen zu kompensieren.

Dabei wird als Basis die Methode der Strukturanalyse von elektrischen und logischen Abhängigkeiten eingesetzt, um die wesentlichen Vorteile der probabilistischen Netzwerke zu nutzen, und gleichzeitig deren Nachteile zu umgehen. Es wird dazu wie bei probabilistischen Systemen der Einsatz von automatisiert ermittelten Abhängigkeiten aus den Entwicklungsmodellen realisiert.

In dieser Arbeit wird jedoch darüber hinaus gehend ein Gesamtansatz betrachtet, der ein umfassendes mechatronisches System mitsamt seinen sich aus der eingesetzten Software ergebenden logischen Abhängigkeiten analysiert. Dabei werden auch die Vorteile der engen Integrierbarkeit in einen Entwicklungsprozess berücksichtigt.

Durch das Vernachlässigen von Ausfallwahrscheinlichkeiten und die Zulassung von Rückkopplungen im System ergeben sich wesentliche Vorzüge gegenüber den traditionellen probabilistischen Netzen. Denn die Erfassung der Ausfallwahrscheinlichkeiten der Strukturelemente ist ein aufwändiger Teil des Vorgehens bei den probabilistischen Netzwerken, der vor allem bei der unüberschaubaren Variantenvielfalt heutiger Systeme nicht ausreichend beherrscht wird. Die Zulassung von Rückkopplungen im System erlaubt die notwendige Betrachtung heutiger vernetzter Systeme mit ihren komplexen Wechselwirkungen.

Aufgrund der beschränkten Menge der Strukturdaten und der Einfachheit des für dieses Vorgehen zu implementierenden Diagnosealgorithmus ist auch der Einsatz auf heutigen Steuergeräten möglich und wird ebenfalls im Folgenden beschrieben.

Mit Hilfe dieser Basismodifikationen an probabilistischen Netzwerken und der einfachen Strukturanalyse soll die erreichbare Diagnosequalität und der dafür notwendige Entwicklungsaufwand gegenüber reinen Strukturanalysen bzw. den probabilistischen Netzwerken verbessert werden. Das für einen solchen Ansatz notwendige Vorgehen wird anhand einer beispielhaften Umsetzung in den folgenden Abschnitten detailliert gezeigt.

5.1.2. Systemstruktur als zentrales Diagnoseelement

Die Erstellung der Systemstruktur umfasst den größten Teil der Wissensbeschaffung der Diagnoselösung. Sie liegt nach Erstellung in Form einer Erreichbarkeitsmatrix der betroffenen Komponenten vor.

Die dort beinhalteten Komponenten sind Informationen aus der Hardware, der Software und der Kommunikation.

Beginnend mit den Strukturdaten werden die elektrischen Komponenten aus einem Stromlaufplan wie in Abbildung 5-1 in Form von Leitungsdaten, Steckern, Steckerbelegungen sowie eingebetteten Komponenten ermittelt.

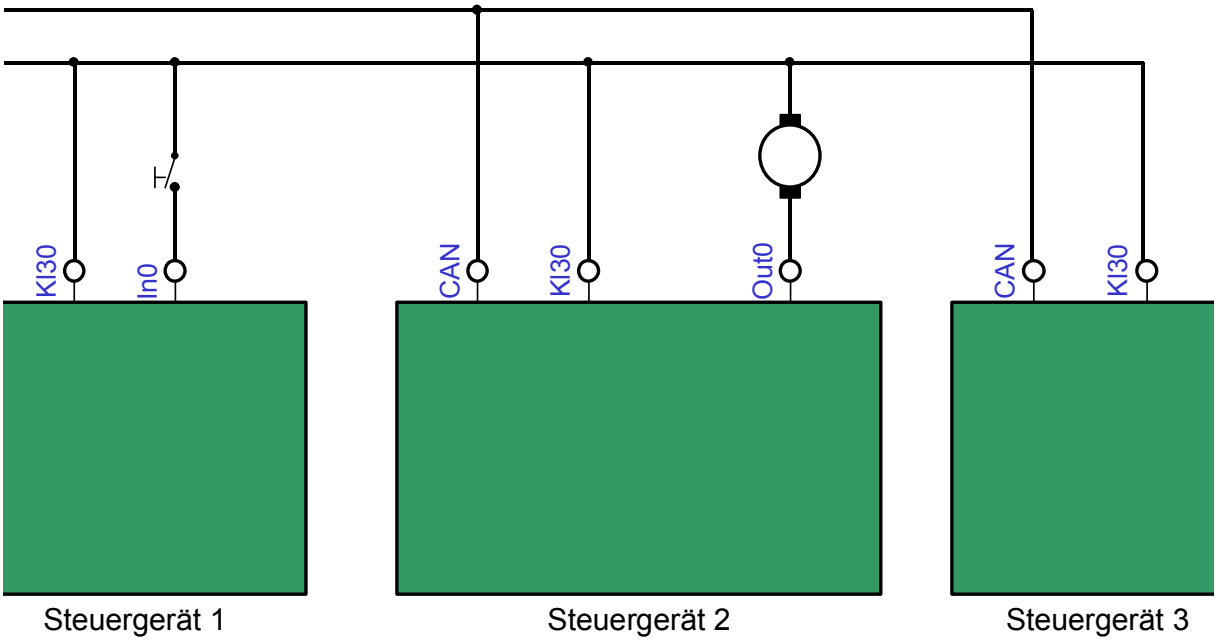


Abbildung 5-1 Schematischer Auszug des elektrischen Schaltplans eines Fahrzeugsystems

Abstrahiert um die elektrische Funktionalität werden diese Komponenten lediglich als Strukturelemente mit gerichtetem Einflussverhalten interpretiert. Die daraus resultierende Systemsicht wird durch Abbildung 5-2 veranschaulicht. Die Pfeile markieren die logische Einflusspfade innerhalb des Systems.

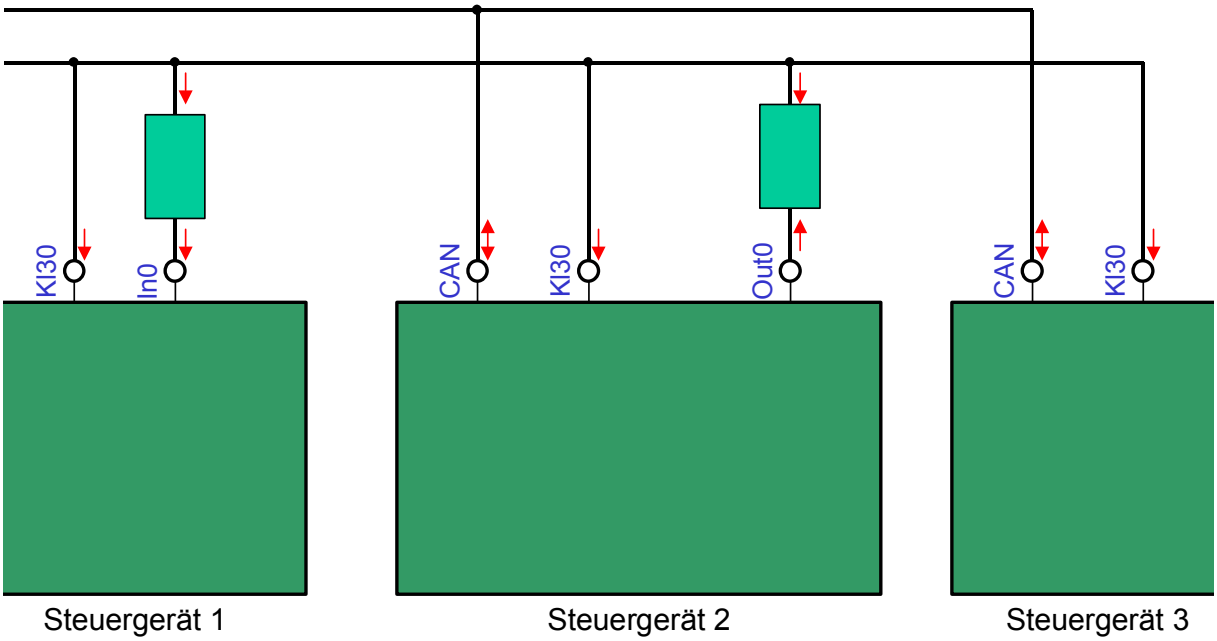


Abbildung 5-2 Abgeleitete Strukturdaten aus der Fahrzeugvernetzung

Auf Basis dieser Strukturen sind bereits erste diagnostische Schlussfolgerungen möglich. Vor allem über die Ausnutzung von Wirkungsketten über gemeinsam genutzte Steckverbindungen kann so effektiv diagnostiziert werden. Solche Verfahren sind bereits in einigen Publikationen vorgestellt worden [Lemm94][Bäke00].

Da aber die Einflussmöglichkeiten von Fehlern in vernetzten Systemen nicht an den Steuergerätegrenzen enden, muss dieses Verfahren erweitert werden. Zu diesem Zweck muss auch die auf den Steuergeräten eingesetzte Software analysiert und in die Systemstruktur eingebracht werden. Im vorliegenden Fall von STEP-X können durch den Einsatz eines modellbasierten Entwicklungsprozesses bereits wesentliche Strukturinformationen aus den grafischen Entwicklungswerkzeugen extrahiert werden. Für die Anwendung in einer strukturbasierten Diagnose ist dabei weniger das Verhalten von Softwarekomponenten interessant, sondern vielmehr wird die Verkettung verfolgt, d.h. der Datenfluss über die Komponenten. Zusammen mit der Definition der Schnittstelle der Software zur Hardwareumgebung des Steuergerätes kann so das bestehende Hardware-Strukturmodell um Softwarestrukturen ergänzt werden. Diese Ableitung und Erweitung der Abhängigkeitsbeziehungen ist in Abbildung 5-3 dargestellt.

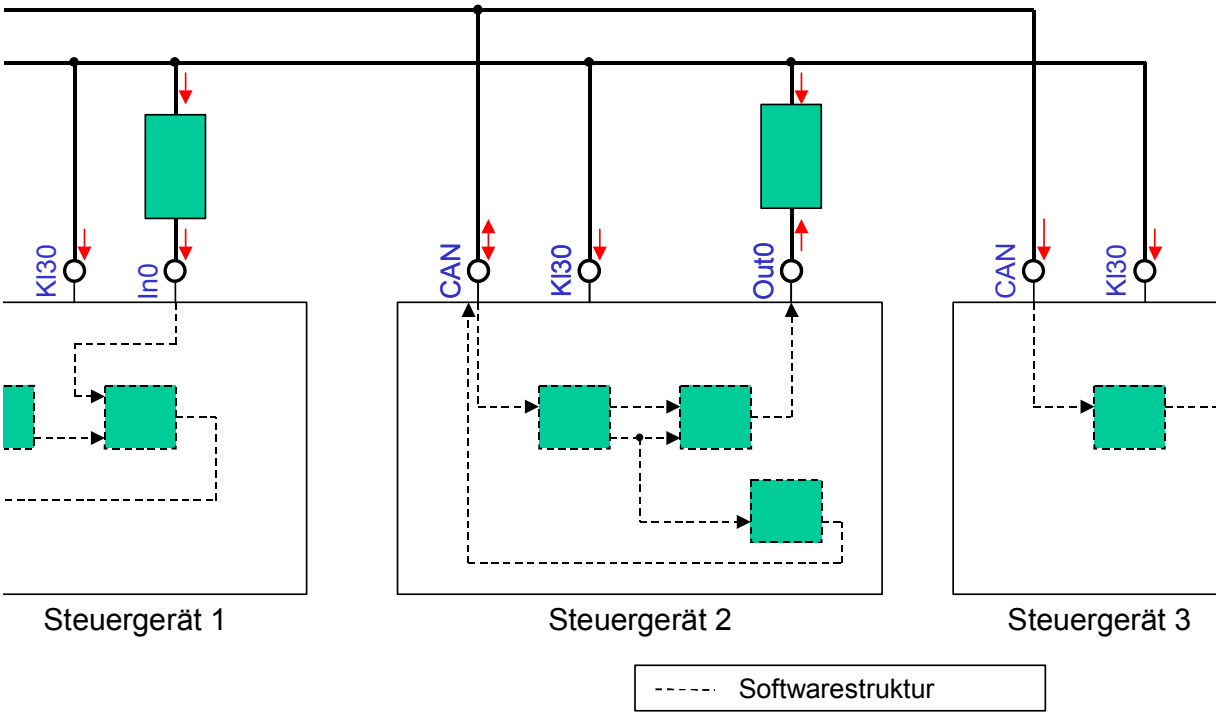


Abbildung 5-3 Systemstruktur ergänzt um Softwarestruktur

Zu sehen ist hier, dass sich z.B. ein Fehler auf dem CAN-Bus über die Software des Steuergerätes 2 auf den Ausgangsport Out0 auswirken kann. Durch die Reduktion des Verhaltensaspektes der verwendeten Softwarekomponenten auf mögliche Auswirkungen unter Vernachlässigung ihres Verhaltens wird die Strukturanalyse der Software stark vereinfacht. Hierdurch lassen sich auch Komponenten in die Analyse integrieren, die lediglich als Black-Box von einem Zulieferer erstellt wurden. Für eine Analyse auf Basis des tatsächlich implementierten Verhaltens wäre die Analyse des häufig nicht vorhandenen Quellcodes der betreffenden Komponente notwendig.

Um die logischen Abhängigkeiten im System noch feiner erfassen zu können, ist parallel zur Softwarestrukturanalyse auch eine Analyse und Integration der Kommunikation im Gesamtsystem möglich. Auf Basis dieser Informationen können Einflüsse von Softwaredefekten und Steuergeräteausfällen für das Gesamtsystem ermittelt werden. Die Erweiterung der bestehenden Systemstruktur aus Hardware- und Softwarestrukturkomponenten um Kommunikationsaspekte ist in Abbildung 5-4 dargestellt. Wichtig ist hierbei wie schon bei der Integration der Softwarekomponenten in die Abhängigkeitsstruktur die klare Kopplung über virtuelle Schnittstellen von Kommunikationssignalen zu Strukturelementen.

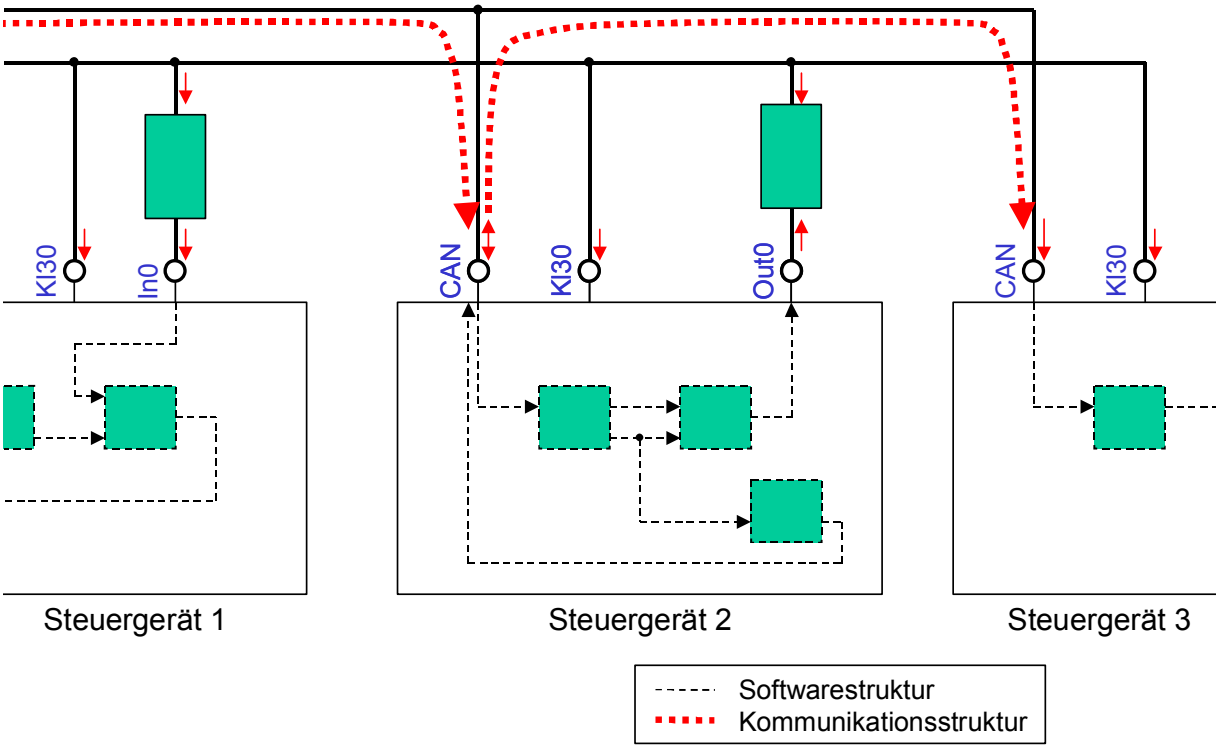


Abbildung 5-4 Systemstruktur ergänzt um Kommunikationsstruktur

Nachdem in den bisherigen Abschnitten dargestellt wurde, dass im Rahmen dieser Arbeit eine Systemdiagnose auf Basis von Strukturinformationen vorgeschlagen wird, soll nun an dieser Stelle konkret beschrieben werden, welche Daten aus dem Entwicklungsprozess bezogen und zu kombinierten Hardware-Software-Strukturen werden. Dazu sind einerseits die Werkzeuge beschrieben, aus denen Strukturinformationen automatisch abgeleitet werden können, und andererseits wird ebenfalls dargelegt, welche Daten die Entwickler zur Realisierung dieses Diagnoseprozesses zusätzlich vorhalten müssen.

5.2. Ermittlung der Strukturdaten

Im Folgenden wird beschrieben, wie die während der Entwicklung zur Verfügung stehenden Informationsquellen zur Erstellung der Diagnosemodelle genutzt werden können. Im Anschluss wird auch die Übertragung der Strukturinformationen zwischen den Steuergeräten im Betrieb und in der Werkstatt thematisiert.

5.2.1. Ermittlung von Strukturinformationen in der Entwicklungsphase

In den vorigen Abschnitten wurde angekündigt, dass sich die vorgeschlagene Vorgehensweise zur Ermittlung von Fehlerkandidaten sehr gut in einen Entwicklungsprozess eingliedern lässt. Dies soll im Folgenden begründet werden.

Die entwickelte Software zur Ableitung von Systemstrukturen während der Entwicklungsphase besteht aus drei prinzipiellen Komponenten:

1. Abhängigkeitsstrukturen aus Softwaremodellen

   Zum Einen wird aus den Softwaremodellen (die Beispielimplementierung basiert auf Modellen in Matlab/Simulink) die zugrunde liegende Abhängigkeitsstruktur ermittelt. Dazu wird für jede Eingangsgröße des Systems verfolgt, auf welche Softwarekomponenten diese Eingangsgröße einen Einfluss hat. Beim Auftreten von Fehlern lassen sich so möglicherweise betroffene Softwarekomponenten leicht identifizieren. Der Strukturbaum mit den abgeleiteten Abhängigkeiten wird im Anschluss mit Hilfe eines Erreichbarkeitsgraphen in eine reduzierte Darstellung umgewandelt, die lediglich noch Ein- und Ausgangsbeziehungen des Steuergerätes beinhaltet.

2. Abhängigkeitsstrukturen aus Hardwarekomponenten

   In DOORS wird die elektrische Struktur des Systems mit allen Variantenmöglichkeiten abgelegt. Dieser Netzplan in DOORS hat den Vorteil, dass er zum Einen menschenlesbar ist, aber dennoch die vollautomatisierte Auslesung durch das Diagnosesystem ermöglicht. Das Resultat nach dem Einlesen von Strukturdaten aus der Hardwarebeschreibung ist eine elektrische Komponentenstruktur des Fahrzeugbordnetzes. Durch die Anreicherung des elektrischen Netzes mit gerichteten Signalinformationen innerhalb der Spezifikation wird dem Diagnosesystem eine erweiterte Schlussfolgerungsmöglichkeit gegenüber der Auswertung der reinen elektrischen Beschaltung gegeben.

3. Abhängigkeitsstrukturen aus Kommunikationsbeziehungen

   Aus den Kommunikationsdaten, die für jedes Fahrzeug in einer so genannten Kommunikations-Matrix abgelegt ist, können zusätzliche Abhängigkeiten in dem System heraus gefiltert werden. Hier wird für CAN-Netzwerke üblicherweise das DBC-Format der Firma Vector Informatik verwendet, und daher auch in der Beispiel-Applikation ausgewertet. Allerdings muss bei der Einbeziehung von logischen Informationen wie CAN-Nachrichten berücksichtigt werden, dass ein Fahrzeug ein sehr komplexes virtuelles Abhängigkeitsnetzwerk auf Basis von Kommunikationsdaten besitzt. In der Praxis zeigte sich daher, dass es notwendig ist, sich mit dem Diagnosesystem auf einige wenige wichtige Botschaften innerhalb des Systems zu beschränken, und so nur die wesentlichen Abhängigkeiten im System abzubilden.

Unter Nutzung einer Schnittstellenbeschreibung können sich diese drei Typen von Abhängigkeitsstrukturen zu einer Systemstruktur vereinen. Diese Schnittstellenbeschreibung erfolgt ebenfalls in DOORS. Sie definiert die Beziehung von logischen Bezeichnern auf Software bzw. Kommunikationsseite mit den Bezeichnern aus dem elektrischen Systemschaltplan, die aufgrund unterschiedlicher Entwicklungs- und Nomenklaturphilosophien nicht kongruent sind. Die so ermittelten Abhängigkeitsstrukturen werden im Anschluss an die Erstellung der Systemstruktur in zwei unterschiedlichen Datenbasen abgelegt: zum Einen aufgeteilt als steuergerätebezogene Daten, in denen jeweils die Strukturdaten enthalten sind, die die unmittelbare elektrische Umgebung eines Steuergerätes beschreiben, sowie die Strukturdaten des Gesamtsystems. Dies ist in Abbildung 5-5 dargestellt.

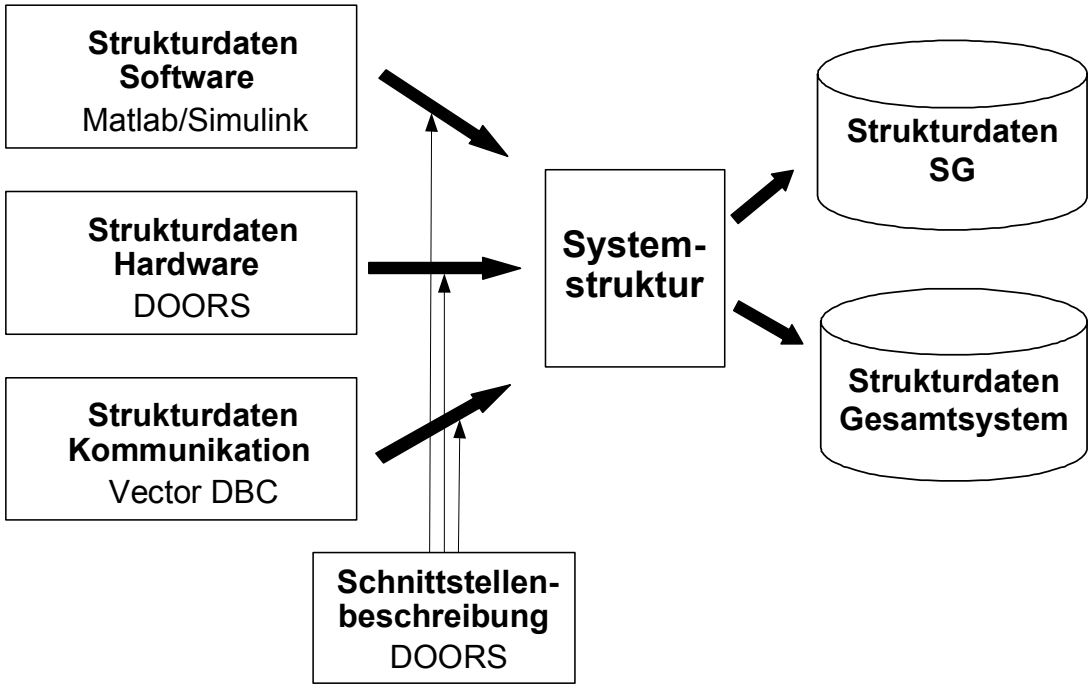


Abbildung 5-5 Kombination von Strukturdaten zu einer Systemstruktur

Anhand dieser vorliegenden Daten können bereits in jedem Stadium der Entwicklung Einflussgrößen von potentiell auftretenden Fehlern bestimmt und für den jeweiligen Einsatzzweck passend begrenzt werden. Minimale Grundlage für die Untersuchungen ist lediglich die Existenz komponentenorientierter Modelle mit implementiertem Daten- und Kontrollfluss.

5.2.2. Ermittlung von Strukturinformationen in der Nutzungsphase

Die Strukturdaten können auch steuergerätebezogen aufgeteilt werden. Dies erlaubt eine flexible Ablage der Daten auch auf jedem einzelnen Steuergerät.

Damit ergibt sich für freie Werkstätten, oder Werkstätten ohne Netzzugang zum OEM der Vorteil, dass direkt mit diesen Daten eine dynamisch erstellte Systemstruktur verwendet werden kann. Im Gegensatz zu den tatsächlich vom OEM herausgegebenen Abhängigkeitsstrukturen für ein ganzes Fahrzeug sind in den Onboard-Versionen nicht die Details der Systemschaltpläne mit allen Splices und Koppelsteckern des elektrischen Kabelbaums zwischen den Steuergeräten enthalten. Dies wird in der Praxis nicht umgesetzt, da eine solche Ablage an Bord des Fahrzeugs erfordern würde, dass bei Integration der Steuergeräte in das spezifische Fahrzeug, die kompletten Strukturdaten auf das Fahrzeug gebracht werden müssten. Bisher werden am Bandende jedoch aus Zeitgründen lediglich minimale Datenmengen für Parametrierungen der Steuergeräte übertragen. Ein Flashvorgang mit einem Datenvolumen, das die Systemstruktur des gesamten Fahrzeugs enthält, wäre daher derzeit nicht wirtschaftlich. Da die Steuergeräte nur ihre lokalen elektrischen Verbindungen in Form von Steckerbezeichnungen, Pinbezeichnungen und den daran angeschlossenen Sensoren und Aktoren aufgrund der Anforderungen des OEMs zur Verfügung stellen können, ergeben diese Daten eine vereinfachte Systemstruktur. Die Zusammenstellung der Systemstruktur auf den Steuergeräten erfolgt durch Kombination der Einzel-Strukturen über deren symbolische Schnittstellenbezeichner.

In Werkstätten mit Zugang zu den Vernetzungsdaten des OEMs für das spezifische Fahrzeug kann auf die dynamische Zusammenstellung dieser steuergeräteinternen Daten verzichtet werden. Hier können entweder über das Netzwerk mit dem OEM oder über Offline-Server-Kopien die für das Fahrzeug gültigen Abhängigkeitsstrukturen zur Verfügung gestellt werden.

Das Ablaufsystem zur Diagnose auf Basis dieser Abhängigkeitsstrukturen ist in beiden Fällen identisch. Lediglich die Granularität der Fehlerlokalisation aufgrund des unterschiedlichen Detaillierungsgrades der Systeminformationen unterscheidet sich.

Im Folgenden wird die Realisierung der dynamischen Erstellung der Systemstruktur schematisch dargestellt.

Die Teilsysteme, d.h. die einzelnen Steuergeräte, diagnostizieren sich wie bisher dezentral selbst durch Ablage konventioneller Fehlercodes. Zusätzlich ist aber auf den Steuergeräten eine Struktur enthalten, die die lokalen Abhängigkeiten des Steuergerätes enthält. Die dabei entstehenden Diagnosen sollen jedoch in einem vereinfachten Gesamtsystemstrukturplan, der in einem Zentral–Modul gehalten wird, zu Systemdiagnosen führen (s. Abbildung 5-6). Der Strukturplan des Gesamtsystems wird somit dynamisch an den jeweiligen Steuergeräteverbund angepasst, und nur bei Erstinbetriebnahme der Steuergeräte übertragen. Damit können

an das Gesamtsystem angepasste Diagnosen trotz unterschiedlichster Steuergerätekonfigurationen, Ausstattungsvarianten und Softwareständen erzeugt werden. Dazu ist die Mitteilung von Steuergerätebeschaltungen und Fahrzeugkabelbäumen von den beteiligten Steuergeräten zu einem übergeordneten Systemdiagnosesteuergerät möglich. Dies kann wie in der Abbildung 5-6 dargestellt, als ein eigenständiges Gerät realisiert werden, oder aber um in der Praxis Kosten zu sparen als zusätzliche Anwendung auf jedem beliebigen Steuergerät mit ausreichend vorhandenen Ressourcen untergebracht werden. Bei der Implementierung eines solchen Vorgehens ist der Vorteil gegenüber statischen Lösungen die jeweils an den konkret vorliegenden Steuergeräteverbund angepasste Systemdiagnose an Bord des Kraftfahrzeugs. Es kommt daher nicht zum Einsatz von großen, evtl. parametrierbaren Matrizen, die den größtmöglichen Nenner aller Systemverbünde umfassen.

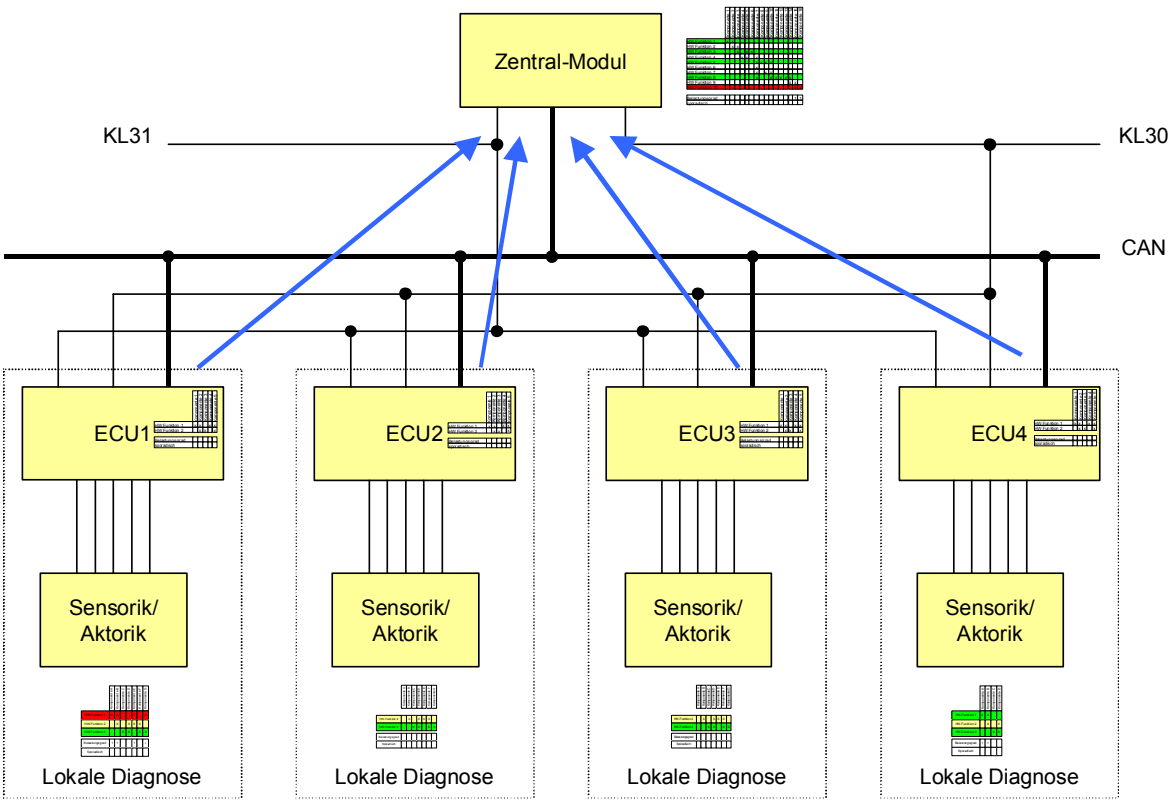


Abbildung 5-6: Lokale und zentrale Diagnose mit Versendung von Abhängigkeitsmatrizen

Ein weiterer wesentlicher Vorteil für eine Onboard-Implementierung dieser Mechanismen ist die Möglichkeit, Systemdiagnosen im Fahrbetrieb zu manifestieren bzw. zu verwerfen. So kann die funktionierende Ansteuerung eines Aktors eine gesamte Abhängigkeitskette an Bord

des Fahrzeugs entlasten, und braucht später in der Werkstatt nicht mehr durch einen Stellgliedtest überprüft zu werden.

Dafür muss die Systemdiagnose auf dem diagnostizierten Fahrzeug integriert sein und zur Laufzeit des Systems den Betrieb mit überwachen. Auch dies wurde im Rahmen der Arbeiten anhand der Beispielanwendung Komfortsystem gezeigt und wird im Abschnitt 7 beschrieben.

5.2.3. Diagnosefunktionen auf dem Steuergerät

Die Erkennung von Fehlern durch die Steuergeräte zur Erzeugung von Fehlerspeichereinträgen erfolgt wie in Abbildung 5-7 schematisch dargestellt. Die eigentlichen Nennfunktionen des entwickelten Steuergerätes operieren getrennt von den Diagnoseumfängen. Die Diagnoseanwendung fungiert als passiver Beobachter und betrachtet lediglich die Hardwareschnittstellen des Systems. Dabei trägt sie beobachtetes Fehlverhalten in den Fehlerspeicher ein. Eine Interaktion zwischen der Diagnoseanwendung und der Nennfunktionen findet nicht statt. Lediglich für die Realisierung von Notlaufverhalten, das in der vorliegenden Anwendung nicht implementiert wurde, wird der Nennfunktion gestattet, Einsicht in den Fehlerspeicher zu erhalten, um die Eintrittsbedingung für einen Notlauf erkennen zu können.

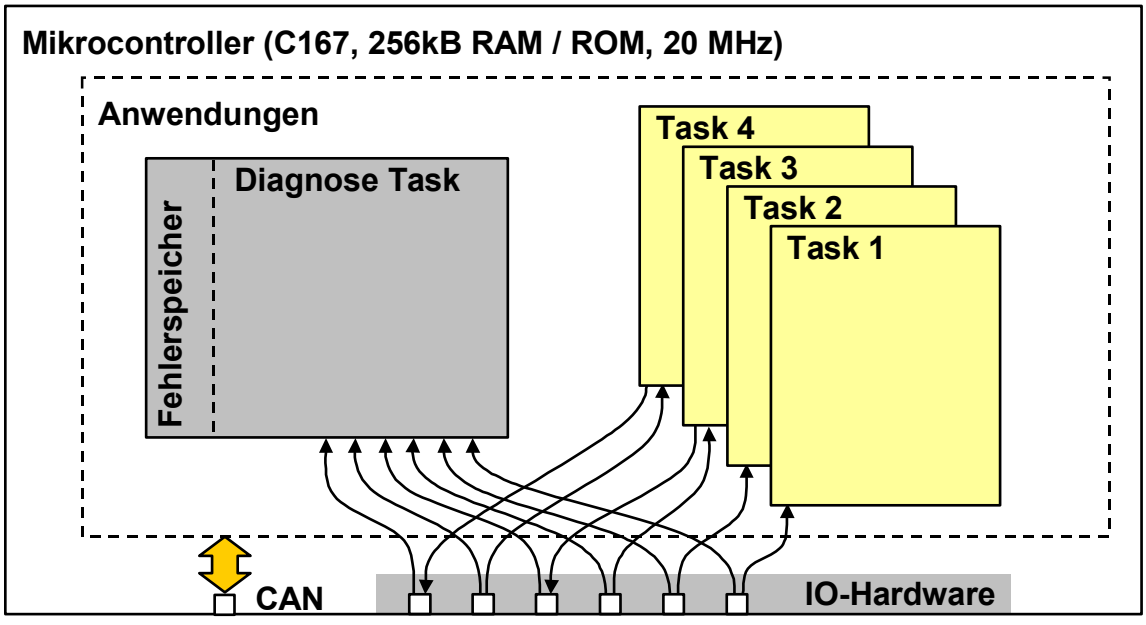


Abbildung 5-7 Realisierung der Diagnoseanwendung auf dem Mikrocontroller

5.3. Basisdaten und ihre Übertragung

Im Fehlerspeicher der Steuergeräte werden die lokal erkannten Fehler abgelegt. Für die Übermittlung der verteilt abgelegten Fehlereinträge zu einem zentralen Diagnosesteuergerät,

wie es in Abschnitt 5.2.2 vorgeschlagen wurde, wird ein für diese Anwendung neu definiertes Übertragungsformat auf Basis des Transport Protokoll 2.0 genutzt [Müll04].

Dazu werden zunächst die im System verfügbaren Strukturabhängigkeiten erstellt. Zusammen mit den vom Steuergerät setzbaren Fehlercodes werden aus Sicht des Mikroprozessors alle diesen Fehlercode potentiell verursachenden Strukturkomponenten ermittelt. Der Extrakt dieser Analyse wird auf den einzelnen Steuergeräten abgelegt. Am Beispiel der elektrischen Spiegelverstellung des Fahrertürsteuergerätes wird ein solcher Extrakt in Abbildung 5-8 dargestellt. Die einzelnen Strukturelemente in den Spalten der Tabelle verfügen zur eindeutigen Kennzeichnung im Netzwerk über eine Identifikationsnummer, die konzernweit eindeutig sein muss um Doppeldeutigkeiten bei der dynamischen Systemerstellung zu vermeiden. Zusätzlich sind wie in der automobilen Diagnose üblich, konzernweit eindeutige Identifikationsnummern für Fehlereinträge in den Zeilenköpfen der Tabelle dargestellt. Ein Kreuz in einem Tabelleneintrag bedeutet für das betroffene Strukturelement in der entsprechenden Spalte, dass es den Fehler, der in der entsprechenden Zeile aufgeführt ist, verursacht haben kann. Damit lassen sich die Abhängigkeitsstrukturen als Netzlisten über die eindeutigen Identifikationsnummern auf den Steuergeräten miteinander verknüpfen und Systemdiagnosen auf den Steuergeräten erstellen.

Zur Realisierung der Struktur- und Fehlercodeübermittlung während des laufenden Kommunikationsbetriebes über den CAN-Bus wurden folgende drei unterschiedliche Transfer-Protokoll-Inhalte definiert:

- Strukturdaten zur Übertragung der Abhängigkeiten
- OK-Meldung zur Meldung von korrekt arbeitenden Funktionen
- Fehlermeldung zur Meldung von Fehlerspeicherinhalten

Die Übermittlung der Abhängigkeitsmatrizen erfolgt über das Transferprotokoll als geschlossene Zeichenfolge über bei der Systemerstellung festgelegte Kanalnummern.

ID		40	41	42	43	44	60	61	62	63	64	65	66	67
		VL Motor Spiegel X	VL Motor Spiegel Y	VL Heizung	VL Motor Spiegel anklappen	VL Stecker SP+Versorgung SG	VL Motorleitung X	VL Motorleitung Y	VL Motorleitung XY	VL Leitung Heizung	VL Motorleitung Spiegel AK +	VL Motorleitung Spiegel AK -	VL Leitung 5V enabled	VL Leitung GND
70	VL Spiegel_X_pos_KS_GND	X					X							
71	VL Spiegel_X_neg_KS_GND/Y	X	X				X	X						
72	VL Spiegel_Y_pos_KS_GND/X	X	X				X	X						
73	VL Spiegel_Y_neg_KS_GND		X					X						
74	VL Spiegel_KS_X/Y	X	X				X	X						
75	VL Spiegel_KS_X/XY	X					X		X					
76	VL Spiegel_KS_Y/XY		X					X	X					
77	VL (Heizung)			X		X				X				X
78	VL (Spiegel anklappen)				X	X					X	X	X	X

Abbildung 5-8 Abhängigkeitsmatrix Spiegelverstellung Fahrertür (VL: Vorne Links)

Um die prinzipielle Stärke des Verfahrens in Form der Onboard-Systemdiagnose am besten ausspielen zu können, müssen im laufenden Betrieb die Statusmeldungen der Steuergeräte häufig ausgetauscht werden. Nur damit erhält die Systemdiagnose ein zeitlich möglichst genaues Prozessbild zur Analyse. Daher werden die Statusmeldungen bei Vorliegen eines Fehlers bzw. bei Erkennung einer Entlastung in einem Raster von zwei Sekunden periodisch übertragen.

5.3.1. Strukturdaten

Die Strukturinformationen werden in einer einzelnen Sendezeile zusammen gefasst und über den CAN-Bus versendet. Das Format zur Übertragung der Daten ist in Tabelle 5-1 dargestellt. Anhand der Anzahl der Spalten und Zeilen der zu übertragenden Strukturinformationen weiß der Empfänger die ankommenden Daten zu Spalten-IDs, Zeilen-IDs und Strukturinformationen korrekt zu interpretieren.

Start-Byte	Typ	Inhalt	Länge
00	Int	Absender-ID	2 Bytes
02	Int	0x0000 (Meldungstyp Strukturdaten)	2 Bytes
04	Int	Anzahl (b) Spalteneinträge	2 Bytes
06	Int	Anzahl (h) Zeileneinträge	2 Bytes
08	Int	Spalten-ID 1	2 Bytes
10..11	Int	Spalten-ID 2	2 Bytes
...	...	...	...
8+(b-1)*2	Int	Spalten-ID b	2 Bytes
10+(b-1)*2	Int	Zeilen-ID 1	2 Bytes
12+(b-1)*2	char[b]	Strukturinfo Zeile 1	b Bytes
13+(b-1)*3	Int	Zeilen-ID 2	2 Bytes
15+(b-1)*3	char[b]	Strukturinfo Zeile 2	b Bytes
...	...	...	...
10 + (b-1)*(h-1)*3	Int	Zeilen-ID h	2 Bytes
12 + (b-1)*(h-1)*3	char[b]	Strukturinfo Zeile h	b Bytes

Tabelle 5-1 Datenformat Strukturinformation über CAN

5.3.2. OK-Meldung

Die OK-Meldung wird von einem Steuergerät versendet, das keinerlei Fehlereinträge verzeichnet hat. Um sicher zu gehen, dass stets der aktuelle Status des Steuergerätes im Systemdiagnosesteuergerät vorliegt, wird auch die OK-Meldung periodisch alle zwei Sekunden übertragen.

Start-Byte	Typ	Inhalt	Länge
00	Int	Absender-ID	2 Bytes
02	Int	0x0001 (Meldungstyp OK-Meldung)	2 Bytes

Tabelle 5-2 Datenformat OK-Meldung

5.3.3. Fehlermeldung

Sobald bei einem Steuergerät Fehlereinträge im Speicher vorliegen, versendet das Steuergerät periodisch alle zwei Sekunden eine Fehlermeldung wie in Tabelle 5-3 Datenformat Fehler-

meldung beschrieben. Die Länge des Telegramms ist dynamisch an die Anzahl der vorliegenden Fehlereinträge und Statusinformationen angepasst. Wenn Fehler vorliegen, wird die zugeordnete Statusinformation als „1“ übertragen. Wenn eine definitive Entlastung eines ursprünglichen Fehlerkandidaten vorliegt, wird als zugeordnete Statusinformation eine „0“ übertragen.

Byte-Nr.	Typ	Inhalt	Länge
00	Int	Absender-ID	2 Bytes
02	Int	0x0002 (Meldungstyp Fehlermeldung)	2 Bytes
04	Char	Anzahl (n) Einträge	1 Byte
05	Int	Zeilen-ID 1	2 Bytes
07	Char	Status-Info 1	1 Byte
08	Int	Zeilen-ID 2	2 Bytes
10	Char	Status-Info 2	1 Byte
...	...	...	...
5+(n-1)*3	Int	Zeilen-ID n	2 Bytes
7 + (n-1)*3	Char	Status-Info n	1 Byte

Tabelle 5-3 Datenformat Fehlermeldung

5.3.4. Anschluss des Testers

Die Interaktion des entwickelten Diagnosesystems mit dem Nutzer in der Werkstatt erfolgt über einen Diagnose-PC. Unabhängig davon, ob der Tester sich auf die Strukturen verlässt, die an Bord des Fahrzeugs ermittelt wurden, oder ob detailliertere Informationen von extern hinzugezogen werden, sind zunächst die Statusinformationen aus den Steuergeräten einzulesen. Dies wurde im Rahmen der Arbeit aus Gründen der flexiblen Anbindung an unterschiedliche Standard-Rechner ohne CAN-Karte als proprietäres Format über die serielle Schnittstelle der eingesetzten Mikrocontroller durchgeführt. Im konkreten Einsatzfall des Diagnosesystems in der Automobilindustrie müsste an dieser Stelle auf die konventionelle Fehlerspeicherübertragung über den CAN oder die K-Leitung zurück gegriffen werden.

Das Verbinden des Tester-PCs an das Steuergerätenetzwerk in der Versuchsanordnung erfolgt über das Steuergerät, dem die Statusinformationen aller Steuergeräte während des Betriebes zugesendet werden. Damit sind an einem Ort die Statusinformation aller Steuergeräte zugänglich. Je nach verwendetem Diagnoseansatz, die Strukturinformationen aus den Steuergeräten zu verwenden, bzw. die Strukturinformationen auf dem Tester zusammen zu stellen, findet

nun auf dem Tester-PC die eigentliche Systemdiagnose bzw. die Anzeige der Fehlerkandidaten statt.

5.4. Schließen von Symptomen auf Fehlerkandidaten

Das Ermitteln von Kandidaten für das Diagnosesystem erfolgt nach einem einfachen Algorithmus, der anhand der Abbildung 5-9 verdeutlicht wird.

Ein Fehlercode bzw. eine Symptombeschreibung belastet zunächst eine oder mehrere Komponenten, an der sich dieser Fehler ausgewirkt hat. Im Beispiel soll das Vorgehen für einen nicht bewegliches elektrisch gesteuertes Fenster im Fond rechts dargestellt werden.

Verdächtig ist natürlicherweise zunächst der elektrische Motoranschluss des Fensterhebers mit der Bezeichnung A6. Über die elektrische Verbindungsleitung zum Pin PWR1_3 am Steuergerät wird die Wirkkette weiter verfolgt. Auch ein Defekt an dieser Leitung erklärt die Beobachtung.

Die Analyse der Software und der Nachrichten auf dem CAN-Bus hat ergeben, dass das Beifahrerfenster einerseits von dem Fensterheberbedienungsknopf an der Tür hinten rechts selbst, aber auch vom Fahrerbedienfeld über den CAN abhängig ist. Daher erweitert sich das Feld der Kandidaten, die dieses Fehlerbild erklären können, um den Fensterhebertaster (Pin A1) samt Anschlussleitung zum Türsteuergerät Hinten Rechts (Pin ANA1_4). Zusätzlich kann ein Defekt auf dem CAN-Bus vorliegen, der die Übermittlung der Fahrerbefehle bzgl. des Fensters verhindert. Durch die Verfolgung der Wirkketten ergibt sich, dass auch ein Defekt beim Bedienfeld des Fahrers an den Pins A3, A5 oder A6 sich auf den elektrischen Fensterheber hinten rechts auswirken kann. An dieser Stelle besteht die Gefahr, dass bei der zu weiten Verfolgung von Signalen durch das System zu viele Fehlerkandidaten gefunden werden. Diese Problematik wird im folgenden Abschnitt 5.4.1 näher betrachtet.

Damit ist die Wirkkette des inoperablen elektrischen Fensterhebers bis zu allen Enden verfolgt, und es gibt keine weiteren Fehlerkandidaten im System.

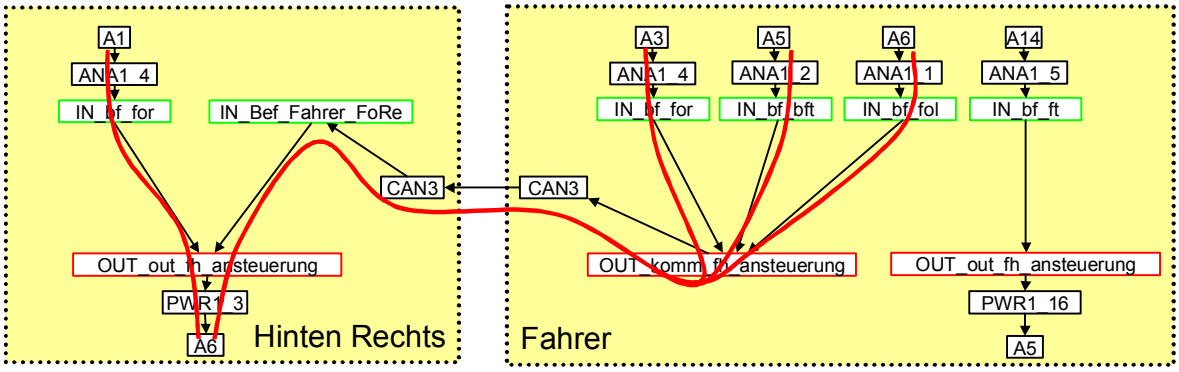


Abbildung 5-9 Ermittlung von Fehlerkandidaten über Abhängigkeiten

5.4.1. Prüfung der Diagnosestrategie)

Die vorgeschlagenen Methoden wurden anhand des STEP-X Versuchsaufbaus in verschiedenen Konfigurationen geprüft. Dabei stellte sich heraus, dass ein einfaches Vorgehen, das lediglich einfache Abhängigkeiten im System verfolgt, keine ausreichende Diagnosequalität erreicht, da die Anzahl zu betrachtenden Fehlerkandidaten für Werkstattzwecke schnell zu groß wird.

Diese Problematik soll an dem Beispiel aus Abschnitt 5.4 verdeutlicht werden. Es fällt auf, dass ein beobachtetes Fehlverhalten an einem Fondsfenster bei vollständiger Betrachtung der Systemabhängigkeiten bereits eine erhebliche Anzahl an Fehlerkandidaten produzieren kann.

So werden in dem vorgestellten Beispiel mit der defekten Fensteransteuerung des Türsteuergerätes „Hinten Rechts“ neben dem Steuergerät „Hinten Rechts“ mit den Fahrer-Bedienfeldern eine ganze Reihe von Objekten des Fahrer-Steuergerätes als Fehlerkandidaten erkannt. Eine Entlastung all dieser Komponenten durch separate Prüfungen während einer Diagnosesitzung in der Werkstatt ist zu aufwändig.

Es mussten daher weitere Einschränkungen bei der Findung von Fehlerkandidaten gefunden werden, um die Fehlersuche in der Werkstatt auf eine mit vertretbarem Aufwand durchführbare Vorgehensreihenfolge zu beschränken.

Es gibt innerhalb des Systems an mehreren Punkten Möglichkeiten, das Verhalten des Strukturdiagnosealgorithmus zu beeinflussen. Dazu zählen im Wesentlichen:

- Die Beschränkung der Anzahl der Komponenten zwischen dem Ort der Fehlerfeststellung und seinen möglichen Fehlerursachen
- Beschränkung des Einflusses der Kommunikation über den CAN-Bus

- Einführung der Entlastung von Fehlerkandidaten

Diese drei Möglichkeiten werden im Folgenden mit ihren Auswirkungen auf die Diagnosequalität vorgestellt.

5.4.1.1. Bewertung der Ausbreitungsweite von Fehlern

Durch die Einschränkung der Ausbreitungsweite auf einen maximalen Wert kann verhindert werden, dass theoretisch mögliche, aber praktisch unwahrscheinliche Fehlerkandidaten zur Anzeige gelangen. Damit wird dem Aspekt Rechnung getragen, dass sich ein Fehler wahrscheinlich in seiner unmittelbaren Umgebung auswirkt und auch dort beobachtet wird.

Die Ausbreitungsweite kann auch als allgemeines Kostenkriterium bei der Berechnung von Fehlerkandidaten aufgefasst werden, wenn die Fehlerkandidaten aufgrund ihrer Distanz zu der den Fehler beobachtenden Komponente sortiert ausgegeben werden (vgl. Tabelle 5-4).

Kosten (Distanz)	Strukturkomponente
0	Motoranschluss Fensterheber
1	Ausgangs-Pin SG
2	Treiber-Ansteuerung SG
3	SW-Eingang Fonds-SG CAN-Signal
4	Analogeingang Fonds-SG CAN-Schnittstelle Fonds-TSG
5	Pin Bedienfeld Fonds-SG CAN-Schnittstelle Fahrer-TSG
6	CAN-Signal
7	SW-Eingang Fahrer-TSG
8	Analogeingang Fahrer-TSG
9	Pin Bedienfeld Fahrer-TSG

Tabelle 5-4 Kandidatenanzeige auf Basis von Fehlerausbreitungskosten

In Abbildung 5-10, das vereinfacht zwei Wirkketten aus dem vorangehenden Beispiel aus Abbildung 5-9 zeigt, wird diese Ausgabe anhand von Ausbreitungskosten verdeutlicht. Die

Ausbreitungskosten werden anhand der Anzahl von Komponenten zwischen der belasteten Strukturkomponente (Motoranschluss Fensterheber) und den einzelnen Komponenten in der Wirkkette berechnet.

Um diesen Effekt zu verstärken, können einzelne Abhängigkeiten mit höheren Kosten, d.h. einer größeren virtuellen Distanz, versehen werden. Genutzt wird diese Möglichkeit im folgenden Beispiel bei der zweitrangigen Betrachtung von Kommunikationsbeziehungen.

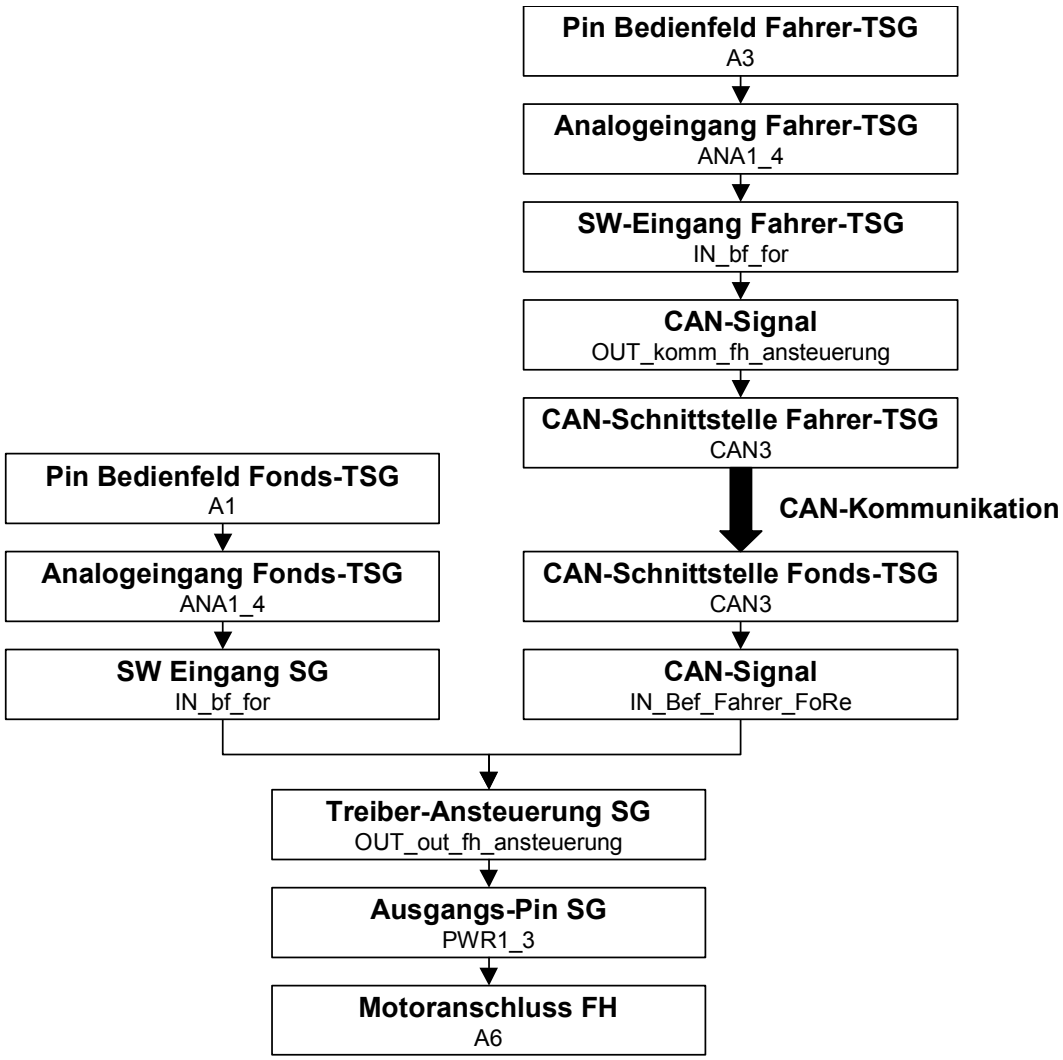


Abbildung 5-10 Ausbreitungswege von Fehlern

5.4.1.2. Bewertung des Einflusses von Kommunikationsbeziehungen

In vielen Fällen hat die Kommunikation über den CAN-Bus weitaus geringeren Einfluss auf das Verhalten des Systems, als es die Abstraktion des Einflusses der analysierten Nachrichten vermittelt. Voraussetzung für einen Serieneinsatz ist, dass die entwickelten Funktionen fehlerfrei implementiert und gründlich getestet sind, daher wird davon ausgegangen, dass Fehler, die sich nur über Kommunikationsbeziehungen ergeben, mit geringerer Gewichtung in dem

Diagnosesystem berücksichtigt werden müssen. Denn durch die in den Steuergeräten hinterlegten Notlauf-Programme ist in vielen Fällen sichergestellt, dass Funktionen auf einem Steuergerät auch dann durchgeführt werden können, wenn ein anderes Steuergerät ausgefallen ist, von dem es abhängig ist. Dabei kann es allerdings zu Einschränkungen kommen, die dem Nutzer auffallen.

Diese Priorisierung lässt sich ebenfalls mit der zur sortierten Ausgabe von Diagnosekandidaten verwendeten Kostenfunktion erreichen, indem Kommunikationsverbindungen mit erhöhten Kosten in die Abhängigkeitsberechnung einbezogen werden. Anhand des vorangegangenen Beispiels ergibt sich mit willkürlich gewählten Kosten von 5 für Buskommunikation die in Tabelle 5-5.

Kosten (Distanz)	**Strukturkomponente**
0	Motoranschluss Fensterheber
1	Ausgangs-Pin SG
2	Treiber-Ansteuerung SG
3	SW-Eingang Fonds-SG CAN-Signal
4	Analogeingang Fonds-SG CAN-Schnittstelle Fonds-TSG
5	Pin Bedienfeld Fonds-SG
8	CAN-Schnittstelle Fahrer-TSG
9	CAN-Signal
10	SW-Eingang Fahrer-TSG
11	Analogeingang Fahrer-TSG
12	Pin Bedienfeld Fahrer-TSG

Tabelle 5-5 Kandidatenanzeige mit Kosten Offset für CAN-Kommunikation

In Abbildung 5-11 wird dieser Effekt anhand der STEP-X-Komfortsystemmodells veranschaulicht. In einer unbedarften Betrachtung werden leicht zu viele Bauteile für eine Prüfung vorgeschlagen. Im Beispiel wird der Fensterheber hinten rechts betrachtet. Der vorliegende Fehler ist eine Leitungsunterbrechung der Zuleitung des Fensterhebermotors. Durch den Feh-

lercode wird zunächst der Fensterhebermotor belastet. Mit zunehmender Distanz von diesem Punkt der ersten Belastung werden zunächst die Steckverbindungen am Motor selbst sowie am Steuergerät als Fehlerkandidaten gefunden. Sind diese nicht als Fehlerursache erkennbar, sind in größerer Belastungsentfernung (im Bild ab Abstand 8) auch die Fensterheberbedientasten und der CAN-Eingang des Steuergerätes hinten rechts mögliche Fehlerursachen. Werden nun bereits Fehlerursachen anhand von CAN-Nachrichten in die Untersuchung einbezogen, steigt die Anzahl von Diagnosekandidaten bereits im Abstand 15 bis auf die für diesen Fehler maximale Anzahl von 38 zu prüfenden Bauteilen an. Werden die Kosten für CAN-Verbindungen erhöht wird dieser Effekt verzögert, und in der Fehlersuche werden vorrangig lokale Diagnosekandidaten betrachtet. Im Beispiel wurden hier willkürlich 25 Abstandseinheiten gewählt, was dazu führt, dass das sprunghafte Wachstum der Kandidatenanzahl erst bei 40 Abstandseinheiten beginnt.

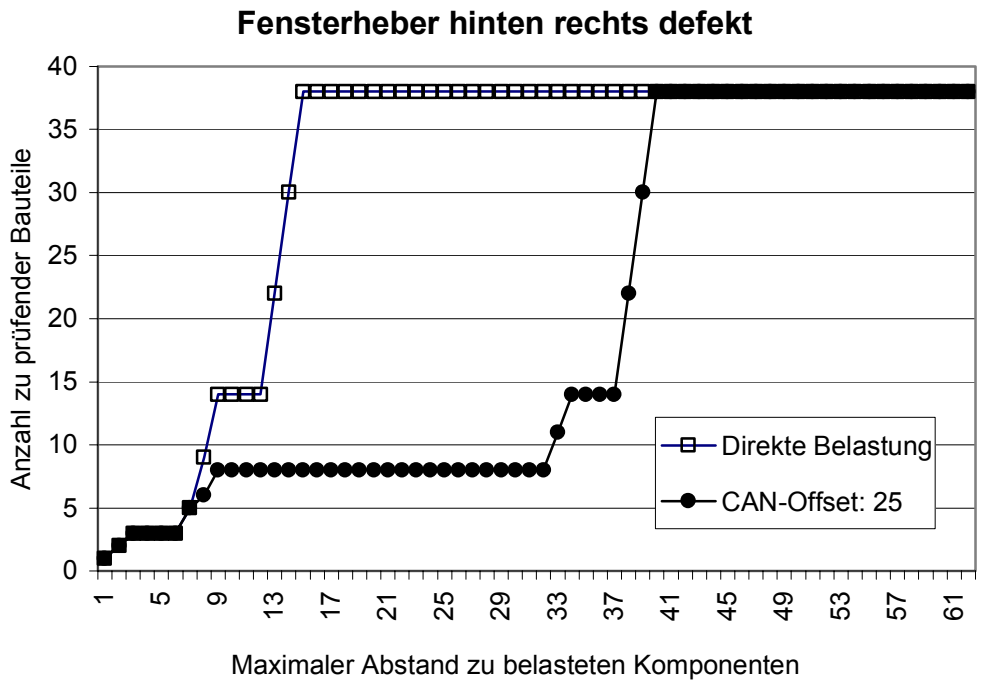


Abbildung 5-11 Anzahl zu prüfender Bauteile bei steigender Distanz vom belasteten Bauteil

In dem Anwendungsbeispiel wurde die nachrangige Betrachtung des Einflusses von Kommunikationsbeziehungen dadurch realisiert, dass die Kosten bei der Abhängigkeitsberechnung für CAN-Nachrichten erhöht wurden. Durch die Beschränkung der Anzahl von Fehlerkandidaten bei der Ausgabe in der Werkstatt werden daher im Regelfall Defekte aufgrund von Kommunikationsbeziehungen nicht ausgegeben. Erst wenn diese erste Liste von Fehlerkandi-

daten als Fehlerursache ausgeschlossen werden konnte und damit die ausgegebene Kandidatenliste erweitert werden muss, kommen solche Kandidaten zur Anzeige.

5.4.1.3. Entlastung von Fehlerkandidaten

Einen wesentlichen Beitrag zur Einsatztauglichkeit des Systems bietet die automatische Entlastung von Fehlerkandidaten durch nachgewiesenes korrektes Verhalten.

Dabei handelt es sich um die zusammenfassende Betrachtung von z.B. Steckverbindungen während der Produktion. Hier kommt es nach Herstellerangaben eher dazu, dass ganze Steckverbindungen nicht gesteckt sind, als dass einzelne Leitungen des Kabelbaums Fehler aufweisen. Mit diesem Wissen kann somit ein Stecker von der Kandidatenliste gestrichen werden, wenn zumindest eine elektrische Verbindung über ihn zustande kommt. Wie in der Abbildung 5-12 dargestellt kann es bei diesem Vorgehen dazu kommen, dass Komponenten an einer zunächst verdächtigen Steckerkomponenten aufgrund einer nachgewiesenen korrekten Steckverbindung ebenfalls entlastet werden. Damit kann die Zahl der Diagnosekandidaten mit steigender Ausbreitungsweite auch fallen.

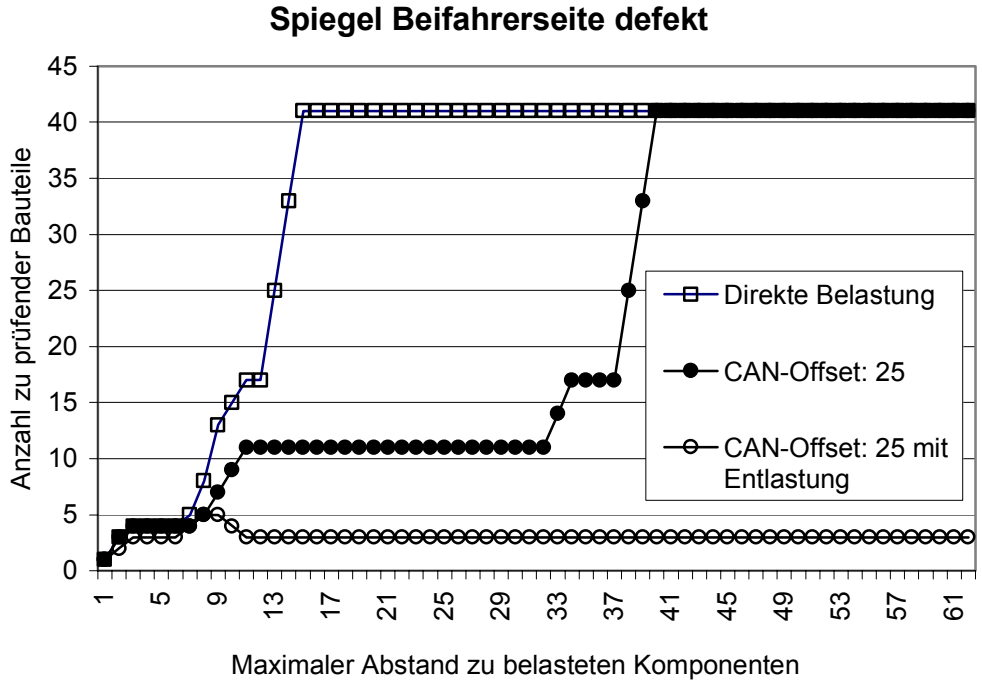


Abbildung 5-12 Auswirkung der Entlastung auf die Anzahl der Diagnosekandidaten

Im Kundendienst muss die Eingrenzungsmöglichkeit durch Entlastung eingeschränkt werden, da wie in Abschnitt 3.2.3.1 aufgeführt, oftmals auch einzelne Kontakte von Steckverbindun-

gen von Fehlern wie Korrosion betroffen sind. Daher kann eine einzelne korrekte elektrische Verbindung über die Steckverbindung somit keinen Rückschluss auf den Zustand der übrigen Kontakte erlauben.

5.4.1.4. Automatische Beschränkung der Anzahl von Fehlerkandidaten

Für die praxisgerechte Nutzung des Systems ist es notwendig, die Anzahl der Diagnosekandidaten zunächst zu beschränken. Erst wenn die Fehlerursache nicht in dieser Auswahl aufgeführt wurde, können die Randbedingungen für die Suche nach Diagnosekandidaten erweitert werden.

In der vorliegenden Arbeit schränkt das Diagnosesystem die Ausbreitungsmöglichkeiten von Fehlern automatisch so weit ein, dass lediglich eine variabel einstellbare Maximalanzahl von Diagnosekandidaten ausgegeben wird. Bei der Berechnung der Diagnosekandidaten wird die allgemeine Kostenfunktion mit den Erweiterungen für CAN-Nachrichten und Entlastungen genutzt.

5.4.1.5. Parametrierung des Diagnosesystems für den praktischen Einsatz

Die Arbeit mit dem Diagnosesystem auf Basis der vorgeschlagenen Strukturanalysen ergibt, dass ein solches System nicht generell und unabhängig vom Anwendungsgebiet einzusetzen ist. Durch die Vielzahl der projektabhängigen Parameter nimmt die Kostenfunktion zur Sortierung der Diagnosekandidaten eine Schlüsselposition bei der Bestimmung der Qualität des Diagnosesystems ein. Mit geeigneten Parameterwerten für die Berechnung der Kosten von Verbindungen innerhalb des Systems und dem Umgehen mit logischen zusammenhängenden Komponenten kann die Aussagequalität gegenüber einer Standardparametrierung deutlich gesteigert werden.

Für die Berechnung der einzelnen Parameter zur Kostenfunktion gibt es keine geschlossene Formel. Die Anzahl sowie der Wert der Parameter ist projektabhängig. Im vorliegenden Fall wurde die Eingrenzung lediglich über Entlastung durch zusammen gehörige Komponenten sowie die zweitrangige Betrachtung von Kommunikationseinflüssen vorgenommen. In anderen Themengebieten als dem hier betrachteten Komfortsystem bestimmen unter Umständen andere Faktoren die Diagnosequalität. Daher muss hier mittels empirischer Versuche in Kombination mit Expertenwissen die für das jeweilige Projekt notwendige Berechnungsvorschrift individuell ermittelt werden.

Im Anschluss an die Ausgabe der Diagnosekandidatenliste muss nach erfolgten Prüfungen in der Werkstatt die Kriterienauswahl gelockert werden, falls die Kriterien die tatsächliche Fehlerursache maskiert haben.

5.4.2. Ablauf der vorgeschlagenen Diagnoselösung

Abschließend wird an dieser Stelle beschrieben, wie das Zusammenspiel der erzeugten Diagnosedaten in der Werkstatt vorgesehen ist. Schematisch ist die Fehleridentifikationskomponente des Diagnosesystems wie in Abbildung 5-13 dargestellt vorgesehen. Beim Besuch des Fahrzeugs in der Werkstatt muss zunächst die Fahrzeugidentifikation über die Diagnosekommunikation ausgelesen werden. Im Anschluss werden die Strukturdaten für die Systemdiagnose zusammen gestellt. Dies kann auf zwei verschiedene Weisen erreicht werden. In einem Fall werden die Strukturdaten von einem zentralen Server bzw. einer Offline-Kopie der Werkstatt durch die Fahrzeug-Konfiguration ermittelt. Alternativ sieht das vorgeschlagene Diagnosesystem vor, die entsprechenden Strukturen in Teilen auf die Steuergeräte zu verteilen. Auf diese Weise ist nicht notwendigerweise eine Serveranbindung bzw. Datenhaltung für Offline-Anwendungen notwendig. Die Aufteilung der Teilstrukturen auf die verschiedenen Steuergeräte erfolgt anhand der lokalen Beschaltung. Elektrische Leitungen, die alleine einem einzigen Steuergerät zugeordnet sind, werden ihm auch direkt zugeschrieben. Anhand von eindeutigen Bezeichnern, die bei der Entwicklung zentral vergeben wurden, können dann Teilstrukturen der Steuergeräte auf dem Werkstattrechner zu einem Gesamtsystem zusammen gesetzt werden.

Im Anschluss an die Erstellung der so ermittelten Systemstruktur werden die eingetragenen Fehlercodes der Steuergeräte ausgelesen, und die Strukturelemente werden im Rahmen der Systemdiagnose belastet. Belastung bezeichnet hier die Markierung von Elementen der Struktur, die durch den Entwickler vorgegeben werden, weil sie für das Auftreten eines Fehlercodes verantwortlich sein können. Durch Anwendung eines einfachen Ausbreitungsalgorithmus, der die Fehlerquellen anhand der Abhängigkeiten in der Systemstruktur traversiert, können weitere Fehlerursachen identifiziert werden.

Nach Identifikation von möglichen Fehlern werden diese mit Hilfe der Werkstatttechnik oder durch Teiletausch behoben, und der Steuergerätestatus wird ein weiteres Mal ausgelesen, um zu überprüfen, ob die Reparatur erfolgreich war. Im Falle von Residuen muss der Diagnosekreislauf noch einmal durchlaufen werden.

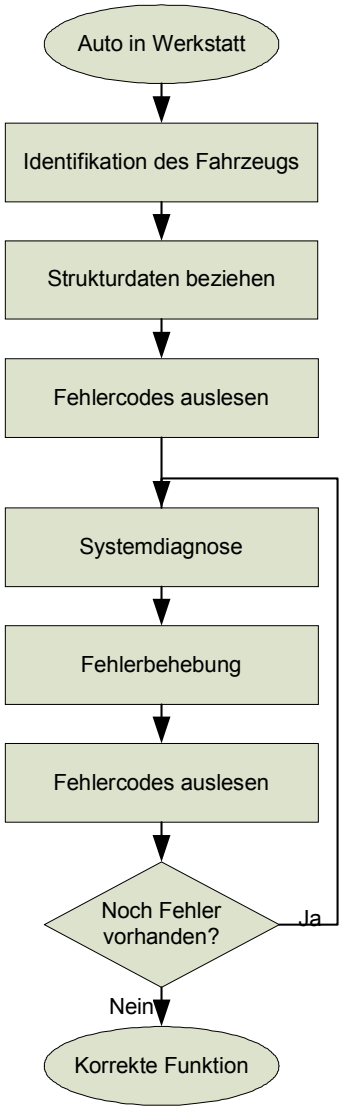


Abbildung 5-13 Prinzipieller Ablauf des Diagnoseverfahrens in der Werkstatt

5.5. Ziele laut Spezifikation für die spezielle Anwendung Komfortsystem

In den für diese Arbeit zugrunde liegenden Spezifikationen zur Entwicklung von Diagnosefunktionen für das Automobil ist vor allem beschrieben, wie mit einmal gefundenen und gespeicherten Fehlern umzugehen ist. Teil dieser Vorgaben zur Fehlerbehandlung ist z.B. die Ablage spezifischer Fahrzeugdaten bei der erstmaligen Fehlererkennung. Diese so genannten Umweltbedingungen sind unter anderem Zeitpunkt, Kilometer-Stand und Momentangeschwindigkeit. Zusätzlich können auch andere Variablen wie die Kühlmitteltemperatur den gespeicherten Daten während der Entwicklung flexibel hinzugefügt werden. Ebenso wird vom Fahrzeughersteller klar definiert, zu welchen Zeitpunkten Fehlereinträge aus dem Fehlerspeicher gelöscht werden dürfen, weil sie als so genannte sporadische Fehler seit langem nicht mehr aufgetreten sind. Auch wird klar definiert, welche Fehler beim vermehrten Auftreten

unterschiedlicher Fehlertypen bei einem sehr kleinem Fehlerspeicher einfach überschrieben und damit ausgeblendet werden dürfen.

Generell wird parallel zum Betrieb des Steuergerätes die ständige Überwachung von RAM und ROM auf Konsistenz verlangt.

Weniger konkret definiert werden in den bisherigen Spezifikationen die Kriterien für die Erkennung von Fehlern, vor allem in nicht sicherheitsrelevanten Teilen des Fahrzeugs, wie dem hier betrachteten Komfortsystem. So wird z.B: gefordert, dass alle Ausgangstreiberstufen mittels einer Transistorschaltung überwacht, und auch alle Eingänge mit Hilfe einer Transistorschaltung zur fallweisen Ansteuerung durch den Mikrocontroller geprüft werden müssen. Gleichzeitig aber werden die konkreten Zeitbedingungen und Stromverläufe für ein Vorliegen eines Fehlverhaltens dem Steuergeräteentwickler beim Zulieferer überlassen.

Notlaufszenarien werden im Rahmen dieser Arbeit als Teil der Nennfunktion des Steuergerätes betrachtet. Im Falle des Auftretens von Notläufen, werden lediglich die eingetragenen Fehlercodes ausgewertet.

6. Einbettung der Diagnose in den Entwicklungsprozess

Vorrangiges Ziel dieser Arbeit ist neben der einfachen Generierung von Diagnosen für ein Fahrzeugsystem die Integration der dafür notwendigen Entwicklungsschritte in den Entwicklungsprozess. Daher wird an dieser Stelle die Methodik beschrieben werden, die Entwickler befolgen müssen, um neben der eigentlichen Funktionsentwicklung auch die Diagnosefunktionen so zu entwickeln, dass sie für die Strukturanalysen geeignet sind.

6.1. Notwendigkeit der Einbettung

Die meisten Diagnoseverfahren für technische Systeme sind auf Basis einer Idee zur Generierung von Fehlerkandidaten entstanden, die aus theoretischer Sicht Erfolg versprechend ist. Dennoch gibt es große Probleme, solche Ansätze in der Praxis mit vertretbarem Aufwand einsetzbar zu machen, weil die Entwicklung der Diagnosefunktionen erst im Anschluss an die Entwicklung und ohne Nutzung der Entwicklungsmodelle stattfindet.

Als Beispiel dafür soll die modellbasierte Diagnose dienen, die in STEP-X im ersten Schritt als Grundlage für die Diagnose im modellbasierten Entwicklungsprozess dienen sollte. Hier wurde zunächst davon ausgegangen, dass ein Großteil der Diagnosemodelle aus den entwickelten Softwaremodellen mit geringem Mehraufwand abgeleitet werden könnte. Bei der Realisierung fiel jedoch auf, dass es bei den für die Diagnose relevanten Modellen nicht um die Steuerungsmodelle der entwickelten Software, sondern vielmehr um die Umgebungsmodelle der Strecke geht, die bis dahin an keiner Stelle entstanden waren. Daher mussten diese Streckenmodelle zusätzlich aufwändig erstellt werden, was den kostengünstigen praktischen Einsatz nur für Anwendungen erlaubt, in denen solche Modelle auch für den Rest der Entwicklung genutzt werden können.

Aus diesem Grund wurde bei der Auswahl des Diagnoseverfahrens großer Wert darauf gelegt, dass das vorgeschlagene Vorgehen neben der Generierung guter Diagnosen vor allem bestehende Entwicklungsdaten nutzt und definiert im Entwicklungsprozess verankert ist.

6.2. Integration in das Anforderungsmanagement

In STEP-X ist die Basis der gesamten Entwicklungskette das Anforderungsmanagement wie in Abbildung 6-1 als umfassender Rahmen um alle Entwicklungsphasen dargestellt [Harm03]. Im Anforderungsmanagement werden Informationen aus allen Entwicklungsphasen gesammelt und verwaltet, um die Realisierung der Anforderungsbeschreibung sicher zu stellen. Aus

diesem Grund sind an dieser Stelle auch alle für die Diagnose relevanten Daten abzulegen bzw. zu extrahieren.

In der Abbildung ist das oben beschriebene V-Modell um die Phasen Architekturenwurf und Partitionierung verfeinert worden, weil aus diesen Phasen wesentliche Informationen für die Diagnose abgeleitet werden können. Das sind zum Einen die Informationen über die verwendete Hardware und die elektrische Vernetzung des Systems aus dem Architekturentwurf sowie die logische Verteilung der Funktionen in der Partitionierungsphase.

Die umfassende Nutzung des Anforderungsmanagementwerkzeuges für das Diagnose- und Testmanagement ermöglicht eine singuläre Datenhaltung in einem Werkzeug. Damit wird die Anzahl der im Entwicklungsprozess eingesetzten Werkzeuge gering gehalten und die Konsistenz der verwalteten Daten erhöht.

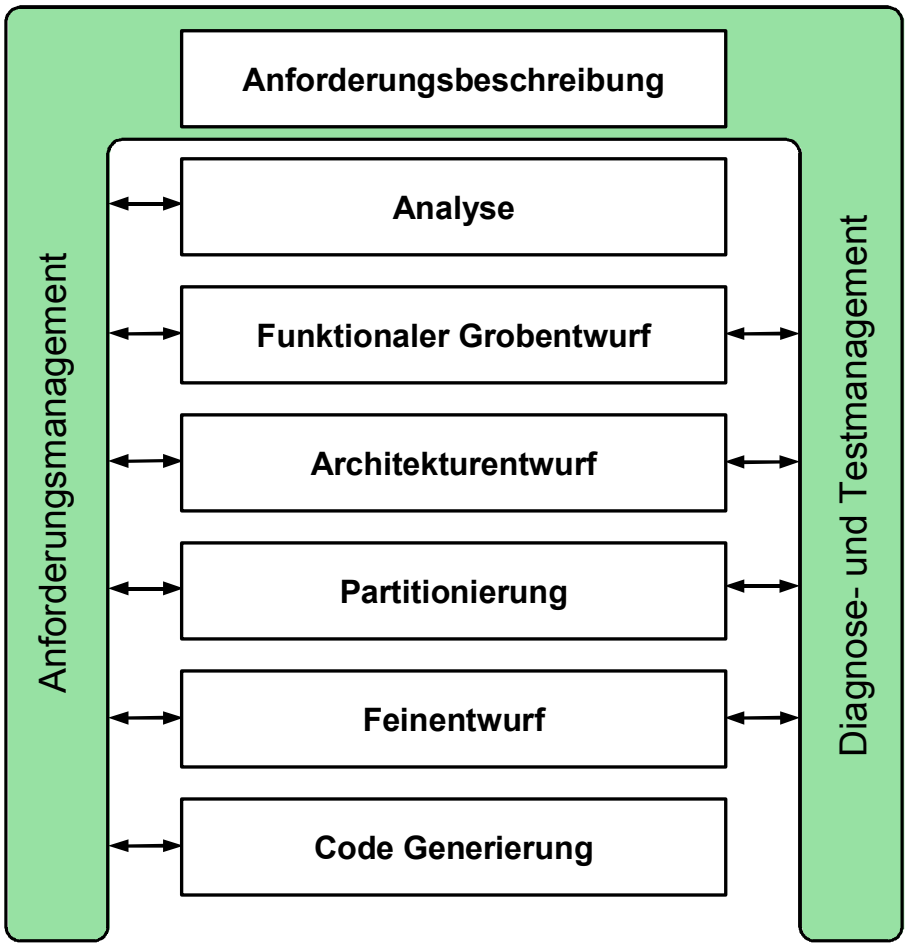


Abbildung 6-1 Anforderungsmanagement als Rückgrat der Entwicklung [Harm03]

6.3. Diagnoseprozess – Wer liefert wann welche Daten?

Im Folgenden wird beschrieben, welche Informationen im Entwicklungsprozess für die Verwendung einer strukturbasierten Diagnose benötigt werden, und ab welchem Zeitpunkt sie im V-Modell zur Verfügung stehen. Dabei wird hier zunächst nur theoretisch beschrieben, welche Daten während der Entwicklung entstehen und in das Requirements-Management-Werkzeug als Ergänzung zur Spezifikation eingetragen werden müssen. Wie dies in einem konkreten Projekt realisiert werden kann, ist Teil der in Abschnitt 7 beschriebenen Fallstudie.

6.3.1. Anforderungsbeschreibung und Analyse

Die Anforderungsbeschreibung enthält die Spezifikation in seiner textuellen Ausgangsform, die für eine automatisierte diagnostische Strukturanalyse nicht zu verwenden ist.

In der Analysephase werden die umzusetzenden Funktionen identifiziert und die Interaktion des Systems mit der Umwelt wird formal beschrieben.

Diese Informationen sind für die hier beschriebenen Vorgehensweisen und Methoden zu grob granular, daher wird erst ab der Grobentwurfsphase eine Vorgehensweise vorgegeben, um die Diagnose zu berücksichtigen.

6.3.2. Grobentwurf

Im Grobenwurf wird das zu entwickelnde System in einzelne Funktionen unterteilt, die auf einer hohen Abstraktionsebene miteinander verknüpft sind. Die hierbei entstehenden Daten beziehen sich auf die logischen Signale, mit denen die Steuergerätesoftware vernetzt wird. Es genügen bereits sehr einfache Angaben aus dieser Phase des Software- und Hardwareentwurfs, um mögliche Fehlerursachen und Fehlerauswirkungen zu bestimmen. Zwar werden an dieser Stelle des Entwicklungsprozesses noch keine pingenauen Bezeichnungen und vielfältige Varianten genannt, dennoch können allgemeine Abhängigkeiten im System abgeleitet werden. Die Information, die sich an dieser Stelle aus dem Entwicklungsprozess entnehmen lässt, hat demnach eine Qualität wie im folgenden Beispiel dargestellt: „Beim Auftreten von Fehlern an der Fensterbedienung am Steuergerät ‚Hinten Rechts' kann die Ursache auch in einer fehlerhaften Implementierung der Auswertung der Fahrertaste für die Fondfenster gesucht werden."

Die Strukturanalyse im funktionalen Grobentwurf ermöglicht es damit bereits wesentlich früher als herkömmliche Diagnoseverfahren, potentielle Fehler im System zu verfolgen. Den

Entwicklern kann man anhand dieser Information gut vor Augen führen, an welchen Stellen im System sich eine Designentscheidung auswirken wird.

Für den Systemdesigner bildet die Interpretation des Grobentwurfs anhand einer Strukturanalyse daher erste Anhaltpunkte für eine im Anschluss an die Entwicklung durchzuführende Fehlermöglichkeits- und Einfluss-Analyse (FMEA). Auch kann hier aufgrund der Analyse-Informationen entschieden werden, dass der Einfluss von bestimmten Signalen im System begrenzt werden muss, um das Gesamtsystem robuster zu gestalten, indem potentielle individuelle Programmierfehler bzw. Hardwaredefekte besser gekapselt werden.

Bereits während des Grobentwurfes wird in der Spezifikation daher die Hardwarestruktur festgelegt, und damit ist bereits in einem frühen Stadium der Entwicklung die elektrische Abhängigkeit der Steuergeräte untereinander abbildbar. Die Daten für diese Struktur werden anhand eines Vorlagenmoduls in DOORS als Text eingetragen, und bleiben damit sowohl automatisch auswertbar, als auch für Menschen lesbar. Auch wenn zu diesem Zeitpunkt der Entwicklung noch die Details der Kabelbäume fehlen, die Zwischenstecker und die Zusammenlegung von elektrischen Leitungen in einem Kabel beschreiben, so ist doch der elektrische Weg von Signalen über die Gehäusestecker der Steuergeräte bereits bekannt. Damit lassen sich bereits wesentliche Systemfehler z.B. über den Totalausfall von Steuergeräten diagnostizieren und örtlich eingrenzen, so dass sich beispielsweise die Auswirkung von abgefallenen Steckkontakten auf das System bestimmen lässt.

6.3.3. Architekturentwurf

Ähnlich gelagert sind die Vorteile im Architekturentwurf. Hier wird die Hardwareschnittstelle des Steuergerätes definiert, der Systemschaltplan erstellt und die Zuordnung von Softwaresignalen zu Pins des Steuergerätes wird vorgenommen. Der Systemschaltplan enthält dabei das elektrische Vernetzungskonzept des zu entwickelnden Fahrzeugs ohne dass dabei bereits Kabelstränge zu einem fahrzeugspezifischen Kabelbaum zusammengefasst werden. Er besteht damit aus Steckern, an denen elektrische Signale geschaltet sind.

Diese Daten werden formal in der Spezifikation notiert und zur automatischen Analyse heran gezogen. Das Format der Einträge zur Hardwarebeschreibung im Architekturentwurf ist in den folgenden Abschnitten festgelegt. Mit diesem Wissen lassen sich nun auch elektrische Defekte bereits im Rahmen der Entwicklung eingrenzen, indem dem Entwickler beim Auftreten von unerwarteten Reaktionen des Systems alle Pins genannt werden können, die eine Funktion beeinflussen. Die fehleranfällige und mit hohem Wartungsaufwand zu betreibende

Verfolgung über nicht formale textuelle Unterlagen zur speziellen Entwicklung entfällt damit vollständig. Die Formalisierung dieser Daten ist neu zum Entwicklungsprozess hinzugefügt worden und bedeutet daher zunächst erhöhten Entwicklungsaufwand. Dieser Aufwand ist aber durch den damit erzielten Zusatznutzen und damit in der Folgezeit verringerten Arbeitsumfängen gerechtfertigt.

6.3.4. Partitionierung

In der Partitionierungsphase werden die Funktionen logisch auf die Steuergeräte verteilt. Diese Phase ist zusätzlich zum Architekturentwurf erforderlich, da es nicht zwingend notwendig ist, dass eine Steuerungsfunktion für einen bestimmten Aktor auch auf dem Steuergerät ausgeführt wird, an den der Aktor angeschlossen ist. Mit anderen Worten kann man ohne Kenntnis der Ergebnisse der Partionierungsphase nicht wissen, welche Steuergeräte an einer bestimmten Funktion beteiligt sind, und demzufolge auch an einem aufgetretenen Fehler beteiligt sein können.

Die Verteilung von Softwarefunktionalität über die Steuergerätegrenzen hinweg resultiert in einem Kommunikationsbedarf zwischen den betroffenen Steuergeräten. Die Summe aller für den Betrieb des Systemverbunds notwendigen Kommunikationsdaten sind in einer so genannten Kommunikationsmatrix abgelegt. Damit beschreibt die Kommunikationsmatrix die Resultate der Partitionierungsphase. Die Kommunikationsmatrix ist bereits in maschinenlesbarer Form notiert und kann daher direkt von der Strukturanalysesoftware ausgewertet werden. Aus diesem Grund wird für die diagnostische Nutzung dieser Informationen kein zusätzlicher Arbeitsaufwand notwendig.

6.3.5. Feinentwurf

Im Feinentwurf werden die im Grobentwurf skizzierten Abläufe und Funktionsumfänge mit allen in der Praxis zu beachtenden Seiteneffekten und Randbedingungen implementiert. Das Resultat des Feinentwurfs ist im Fall von STEP-X ein grafisch notiertes Modell der Software der entwickelten Steuergerätefunktionen in Verbindung mit der elektrischen Vernetzung des Systems.

6.3.5.1. Signalbeschreibung für die Software

Bei der Entwicklung der Nennfunktionalität werden in den vorgelagerten Entwicklungsphasen zunächst abstrakte Schnittstellensichten verwendet, die von idealen und rauschfreien Eingangssignalen ausgehen. Dies ist in Abbildung 6-2 anhand der Kapselung der Nennfunktiona-

lität durch Ein- und Ausgangsdatenverarbeitung durch Hardware- und Softwaretreiber dargestellt. Sowohl Test als auch Diagnose erfordern daher die Erweiterung der anwendungsbezogenen Modelle um Aspekte aus der Systemumwelt. Dazu sind die genaueren Einflussfaktoren der Umgebung (wie EMV, Rauschen, Prellen von Kontakten etc.) auf einzelne Signale sowohl quantitativ als auch qualitativ zu beschreiben, um zu stabilen und auswertbaren Eingangswerten für die Modelle zu kommen. Daher werden im Feinentwurf auch die Schnittstellenfunktionalitäten zwischen der Nennfunktion und der physikalischen Verbindung zwischen Steuerung und mechanischer Umgebung entwickelt.

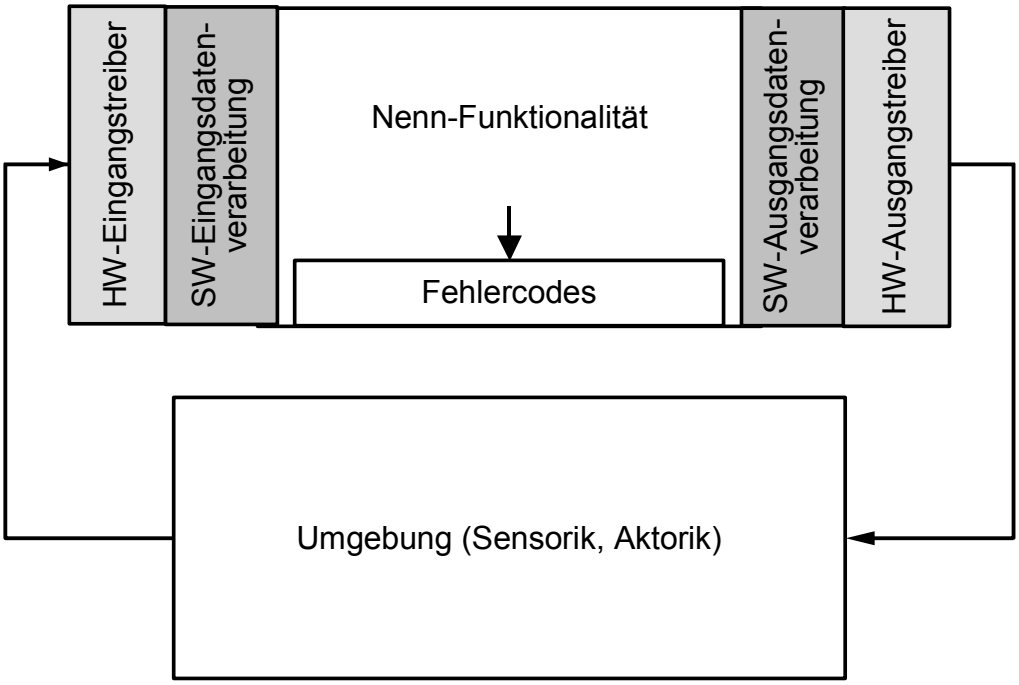


Abbildung 6-2: Funktionale Softwareanteile im Steuergerät

Die Integration dieser Schnittstellenbeschreibung ist essentiell zur Einbindung der Diagnose in den Entwicklungsprozess, da die Diagnose während des regulären Betriebs des Steuergeräteverbundes eine korrekte Softwarefunktionalität voraussetzt und sich auf die Erkennung von Fehlern anhand der o. g. Schnittstellen konzentriert. Diese Informationen sind auch für den Test nützlich. Besonders im Hinblick auf die Wiederverwendung von Tests auf verschiedenen Abstraktionsebenen stellen diese Vorgaben eine Vereinfachung für die Testinstrumentierung dar.

In der Spezifikation wird die Beschreibung der notwendigen Schnittstellen im Modul Systemdaten vorgenommen. Dieses Modul wird zusammen mit der Erläuterung der Umsetzung des Anwendungsbeispiels im Abschnitt 7 näher beschrieben.

6.3.5.2. Kopplung Hardware - Software

Durch die Nutzung grafischer Beschreibungsmittel kann der Datenfluss über die eingesetzten Softwarekomponenten automatisiert verfolgt werden. Damit lassen sich aus Sicht der Diagnose weitere Strukturabhängigkeiten eingrenzen. Am Beispiel von Steuerungsmodellen in Matlab/Simulink hat sich im Rahmen dieser Arbeit gezeigt, dass sich die zugehörigen Abhängigkeiten der Komponenten untereinander automatisiert ableiten lassen.

In der Spezifikation ist für die Verfolgung von Signalen über Steuergerätegrenzen hinweg eine Rubrik eingeführt worden, die Softwareschnittstellen mit im Schaltplan existenten Pins des Steuergerätes assoziiert. Ohne diese Verknüpfung lassen sich keine Abhängigkeiten über Steuergeräte hinweg verfolgen. Damit ist an dieser Stelle ein weiterer zusätzlicher Arbeitspunkt für die Entwickler der Steuergeräte hinzugekommen: es wird gefordert, Steuergerätesoftwareschnittstellen mit Steuergerätehardwareschnittstellen in der Spezifikation zu verbinden. Bisher lag dieses Wissen implizit in den Beschreibungen der Steuergerätehardware- und Steuergerätesoftwareschnittellen. Damit entsteht kein zusätzlicher Mehraufwand bei der Entwicklung, es wird lediglich gefordert, dass die Datenbeschreibung für Schnittstellen formal in der Spezifikation stattfindet.

6.3.5.3. Elektrisches System

Bezogen auf die Entwicklung des Kabelbaums eines Fahrzeugs entstehen im Feinentwurf alle ausstattungsspezifischen Varianten. Diese detaillierten Informationen werden in Datenbanken und Schaltplanverwaltungswerkzeugen abgelegt. Eine Integration dieser Detaildaten in die Spezifikation ist für den Zweck der Entwicklung von Diagnosefunktionen nicht angemessen. Daher wurde für die Verwendung von CAD-Daten des Kabelbaums ein Importwerkzeug für das Diagnosesystem realisiert.

6.3.6. Codegenerierung

In der Codegenerierungsphase wird der grafisch entwickelte Systemfunktionsumfang in eine Hochsprache wie z.B. C oder C++ übersetzt. Dieser Quellcode kann dann für die jeweiligen Steuergeräte kompiliert werden. Die Umwandlung erfolgt direkt, so dass das Resultat der Codegenerierungsphase bei als fehlerfrei vorausgesetzter Übersetzung aus Sicht des Verhaltens mit dem Feinentwurf übereinstimmt. Damit bietet die Codegenerierungsphase aus Strukturanalysesicht keine weiteren Informationen zur Eingrenzung von Fehlverhalten.

6.4. Diagnosetiefe

Bereits in der Spezifikation wird für das Diagnosesystem vorgeschrieben, welche Diagnosetiefe erreicht werden soll. Damit wird z.B. für den Werkstattbetrieb festgelegt, welche Strukturelemente zu einer kleinsten tauschbaren Einheit, wie z.B. einem Steuergerät, gehören. D.h. die Forderung nach der Diagnosetiefe kann pro Funktionalität und Sensor- bzw. Aktorkomponente bereits in die Spezifikation geschrieben werden.

Unabhängig von der Diagnosetiefe der Ausgaben des Diagnosesystems bedingen aber die Eingangsdaten für die Strukturerstellung, dass eine pingenaue und auch softwarekomponentengenaue Diagnose in jedem Fall möglich ist. Bei der Vorgabe einer geringen Diagnosetiefe werden daher bis in eine große Diagnosetiefe Fehlerkandidaten ermittelt, aber bei der Ausgabe werden lediglich die entsprechend der vorgegebenen Diagnosetiefe zulässigen Fehlerkandidaten aufgeführt.

In diesem Zusammenhang ist bei der Angabe der Diagnosetiefe auch mit anzugeben, in welcher Form elektrische Fehler angenommen werden sollen. Es ist z.B. vorstellbar, zunächst nur Steckerfehler an Leitungsverbindungen anzunehmen und erst in einem zweiten Schritt die Diagnosetiefe zu vergrößern, und einzelne Leitungsverbindungen als Fehlerkandidaten anzugeben.

Das vorgeschlagene Diagnosesystem erlaubt die Fehlersuche in einem System bis auf unteilbare Komponenten, wobei die notwendigen Daten direkt aus dem Entwicklungsprozess abgeleitet werden können. Die zur Verfügung stehende Rechenleistung für die Ermittlung der Diagnosen ist auf einem Tester-PC für die vorgeschlagenen Algorithmen ausreichend.

Die Speicherung der Strukturdaten aller möglichen Fahrzeugvarianten auf jedem Werkstattrechner scheint aufgrund der Menge der regelmäßig vom OEM mit aktualisierten Daten nicht angemessen. Da die fahrzeugspezifischen Strukturdaten nur beim tatsächlichen Arbeiten mit einem Fahrzeug benötigt werden, kann auch über eine Online-Verbindung zum OEM die Abhängigkeitsstruktur bezogen werden.

Als Engpass für die Werkstätten könnte sich allerdings in diesem Fall der Zugang zu den Vernetzungsdaten in einer Online-Verbindung zum OEM darstellen. Daher sieht der vorgeschlagene dezentrale Vernetzungsansatz mit den Strukturinformationen direkt auf den Steuergeräten vor, die benötigten Informationen direkt aus dem betroffenen Automobil zu erlangen. Damit ist eine direkte Verbindung der Werkstatt zu den Servern mit den Vernetzungsdaten des OEM nicht immer erforderlich.

6.5. Generierung aus dem Entwicklungsprozess

Gemäß der Intention von STEP-X, einen durch alle Werkzeuge schlüssigen und durchgängigen Entwicklungsprozess zu schaffen, wurden auch die Diagnosedaten in die für STEP-X gewählten Standardwerkzeuge des Entwicklungsprozesses integriert und nur bei explizitem Diagnosewissen speziell für die Diagnose erweitert. Einen Überblick über das generelle Vorgehen und die Anbindung an den Entwicklungsprozess wird im Folgenden gezeigt.

6.5.1. Struktur der Hardware

Die Wissensakquisition für die Diagnose beginnt bei der Verwendung von Hardwarestrukturen. Diese können in der Entwicklungsphase zunächst grob innerhalb der Spezifikation abgelegt und automatisch extrahiert werden, wenn diese anhand des hier vorgeschlagenen Formates vorliegen.

Während der Produktion und im Feld können für ein spezifisches Fahrzeug die Vernetzungsdaten anhand der verbauten Teilenummern des Kabelstrangs genutzt werden. Im von STEP-X betrachteten Anwendungsbeispiel für Volkswagen konnte gezeigt werden, dass die volkswageneigenen Systeme zur Datenhaltung der elektrischen Verbindungen zur Erstellung des kompletten Systemschaltplans genutzt werden können.

Den unterschiedlichen Wissensstand zwischen Systemschaltplan in der frühen Anforderungsdefinitionsphase und dem späteren individuellen Fahrzeugschaltplan dokumentiert Abbildung 6-3. In frühen Entwicklungsphasen existiert nur ein Systemschaltplan, der die elektrischen Verbindungen logisch zwischen den Steuergeräten beschreibt. Diese Information wird für das Projekt STEP-X innerhalb von DOORS als Anhang zur Spezifikation gepflegt, da Zugänge zum Volkswagen-Netz fehlten. Aber die vorgeschlagene Nutzung eines Teils der Spezifikation zur Definition elektrischer Verbindungen hat auch den Vorteil, dass alle an der Entwicklung beteiligten Personen auf einer gemeinsamen Datenbasis arbeiten. Bisherige Lösungen mit getrennt verwalteter Spezifikation und Systemschaltplan haben unterschiedliche Entwicklungswerkzeuge und Datenbasen, in denen die elektrische Daten abgelegt sind. Dies kann zu Konsistenzproblemen führen.

Zu diesem frühen Zeitpunkt gibt es noch keine ausgeprägten Kabelbäume, die zwischen Fahrzeugvarianten unterschiedlich sind, und mit Hilfe von Koppelsteckern unterschiedliche Teilbäume verbinden. Diese Daten werden erst mit Erstellung des Kabelschaltplans im Feinentwurf der Entwicklung erstellt. In der Feinentwurfsphase wird die Datenhaltung der elektrischen Verbindungen aufgrund ihres Umfangs nicht mehr in der Spezifikation gepflegt, son-

dern in den speziell dafür entwickelten Werkzeugen. Da zu diesen Werkzeugen der Zugriff auf das Volkswagen-Netz fehlte, wurde anhand eines Golf-Beispiels lediglich ein einzelnes Fahrzeugprojekt betrachtet, der aus diesem Werkzeug extrahierte Daten enthielt. Das STEP-X Beispiel wurde wie im Grobentwurf in der Spezifikation weiter gepflegt.

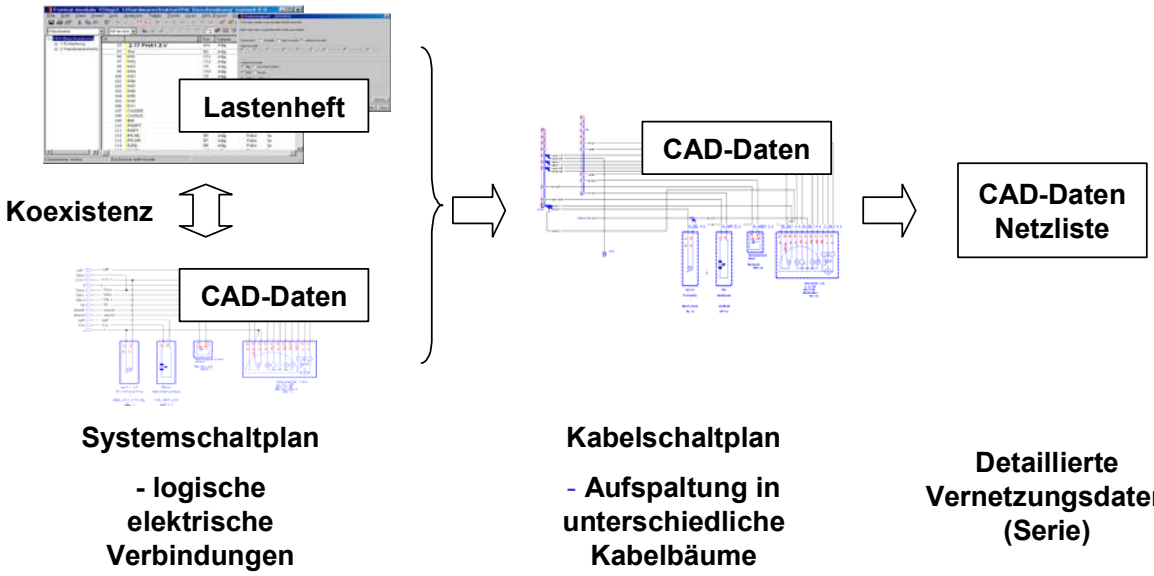


Abbildung 6-3 Datenhaltung während Entwicklung und Produktion

6.5.2. Struktur der Kommunikation

Bereits aus den ersten Untersuchungen des Diagnoseverfahrens an Daten aus realen Fahrzeugen ergab sich, dass allein die Kenntnis der Hardwarestruktur noch nicht in ausreichendem Maß zu eindeutigen Diagnosen führt. Bei Fehlern in der CAN-Kommunikation bestimmter Geräte, die für systemweit relevante Nachrichten zuständig sind, ergeben sich mit diesem Vorgehen allein nicht interpretierbare Fehlermuster. Im vorliegenden Fall war z.B. das Bordnetzsteuergerät eines Volkswagen Golf durch das Abziehen seines Massesteckers als defekt im Fehlerspeicher eingetragen. Durch die Abhängigkeit der Diagnosefunktionalitäten anderer Steuergeräte von einer bestimmten Nachricht des Bordnetzsteuergerätes (in diesem Fall: Klemme 15 ein, d.h. Zündung an) wurden diese ebenfalls als defekt erkannt, da sie bei Ausbleiben dieser Nachricht keine Diagnosekommunikation mehr erlaubten.

Aus diesem Grund wurde die Kommunikationsmatrix der Steuergeräte untereinander in die Systemstruktur eingearbeitet. Betrachtet werden in diesem Fall ausschließlich die von Fachleuten als relevant eingestufte Nachrichten. Eine Verwendung aller Nachrichten innerhalb des Systems würde die Aussagekraft der Kommunikationsstruktur abschwächen, da Untersuchun-

gen an den Kommunikationsbeziehungen unterschiedlicher Fahrzeugplattformen gezeigt haben, dass fast jedes Steuergerät von allen anderen Steuergeräten des Systems indirekt abhängt.

Wie die Hardwarestrukturinformationen auch, werden die Kommunikationsbeziehungen richtungsabhängig in die Abhängigkeitsmatrix eingepflegt. Soll laut Kommunikationsmatrix ein Steuergerät von einem anderen ein relevantes Signal empfangen, dann gilt dieses Steuergerät als abhängig von dem sendenden Steuergerät, während das sendende Steuergerät als unabhängig vom empfangenden Steuergerät betrachtet wird.

6.5.3. Struktur der Software

Völlig ohne Arbeitaufwand für den Entwickler werden die Abhängigkeiten der entwickelten Steuergerätesoftware aus den Entwicklungsmodellen abgeleitet. Dabei wird lediglich das Modell ausgewertet, das für die Codeerzeugung auf dem Steuergerät vorgesehen ist. Damit ist sicher gestellt, dass die Abhängigkeitsbeziehungen, die innerhalb der Diagnose ausgewertet werden, auch diejenigen sind, die in das Fahrzeug eingebracht werden. Eine neue Version der Steuergerätesoftware bedingt somit auch immer eine neue Version der zugehörigen Strukturableitungen für die Systemdiagnose.

Für die im Rahmen der Dissertation betrachteten Beispiele wurde die Feinentwurfs-Software für die Steuergeräte innerhalb von Matlab-Simulink entwickelt, und anschließend mit einem Codegenerator in Quellcode für die Steuergeräte übersetzt. Daher wurde eine Strukturanalysekomponente in das Diagnosesystem eingearbeitet, das auf Basis von Simulink-Dateien die darin enthaltenen Funktionsabhängigkeiten extrahiert. Für die Ableitung der Funktionsabhängigkeiten ist keine Lizenz der Matlab-Simulink-Entwicklungsumgebung notwendig, da die Analysen unabhängig davon auf den statisch daraus erzeugten Modelldateien arbeiten kann.

Für Steuergeräte, die vom OEM selbst entwickelt werden, ist die Abhängigkeitsanalyse aufgrund der guten Verfügbarkeit des Quellcodes leicht gegeben. Meist ist jedoch der OEM lediglich Auftraggeber für ein komplettes Steuergerät, auf dessen Software-Quellen kein Zugriff besteht. Für diesen Fall wird vorgeschlagen, die Abhängigkeitsstrukturen direkt beim Zulieferer durch die Analysesoftware ermitteln zu lassen. Dabei wird sicher gestellt, dass keine Urheberrechte verletzt werden, da die ermittelten Abhängigkeiten lediglich die Abhängigkeiten zwischen Ein- und Ausgängen des Steuergerätes widerspiegeln und nicht das Verhalten der Software.

Am Simulink-Modellbeispiel aus Abbildung 6-4 wird die Ermittlung der Abhängigkeiten aus der Software Schritt für Schritt durchgehend erläutert. Das Modell besteht aus jeweils drei

Modellein- und Ausgängen, die für die Strukturanalyse mit dem Präfix IN_ bzw. OUT_ gekennzeichnet wurden. Die eigentliche Signalverarbeitung erfolgt in den Subsystemen A und B.

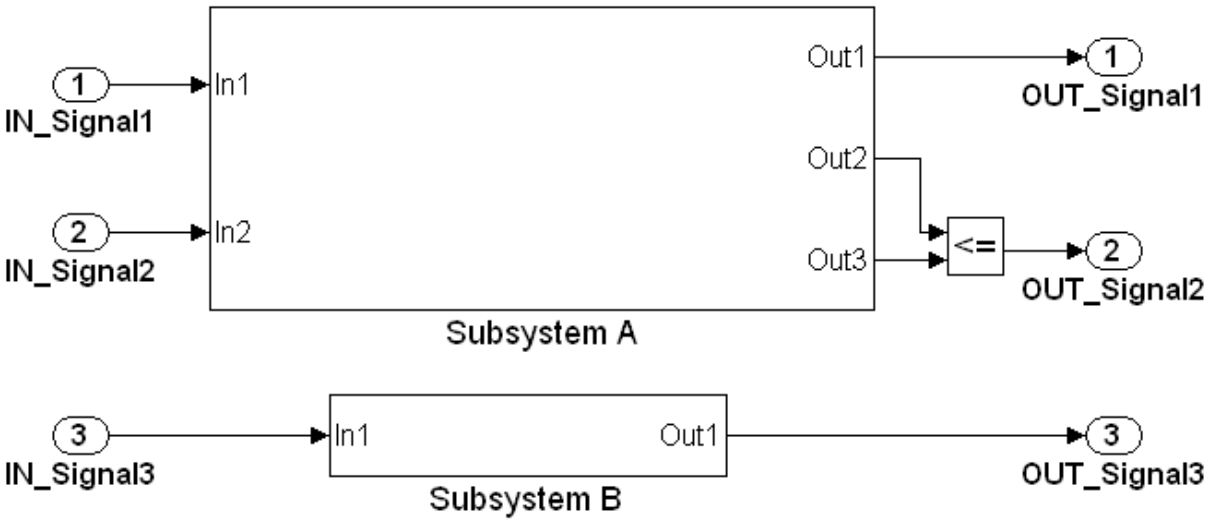


Abbildung 6-4 Simulink Modellbeispiel

Zunächst wird die hierarchische Struktur des Modells aufgehoben, um die Signalpfade besser in einem Bild darstellen zu können. Das Ergebnis ist in Abbildung 6-5 zu sehen.

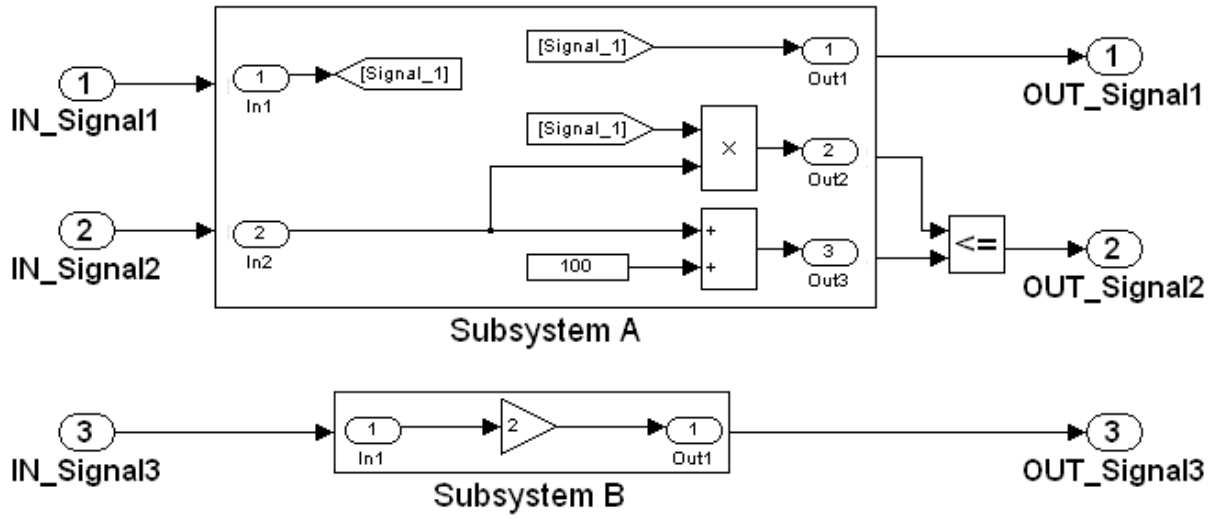


Abbildung 6-5 Simulink-Modellbeispiel - Hierarchien aufgelöst

Im Folgenden werden alle symbolischen Verweise und Signalauswahlmechanismen von Simulink in Form z.B von From/Goto-, Selektor-, Bus Creator-, und Bus Selektor-Blöcken ersetzt durch direkte Verbindungen der Komponenten untereinander. Im Beispiel in Abbildung

6-6 wird die symbolische Verwendung des Signalnamens Signal_1 in einer Goto/From-Kombination durch direkte Verbindungen ersetzt.

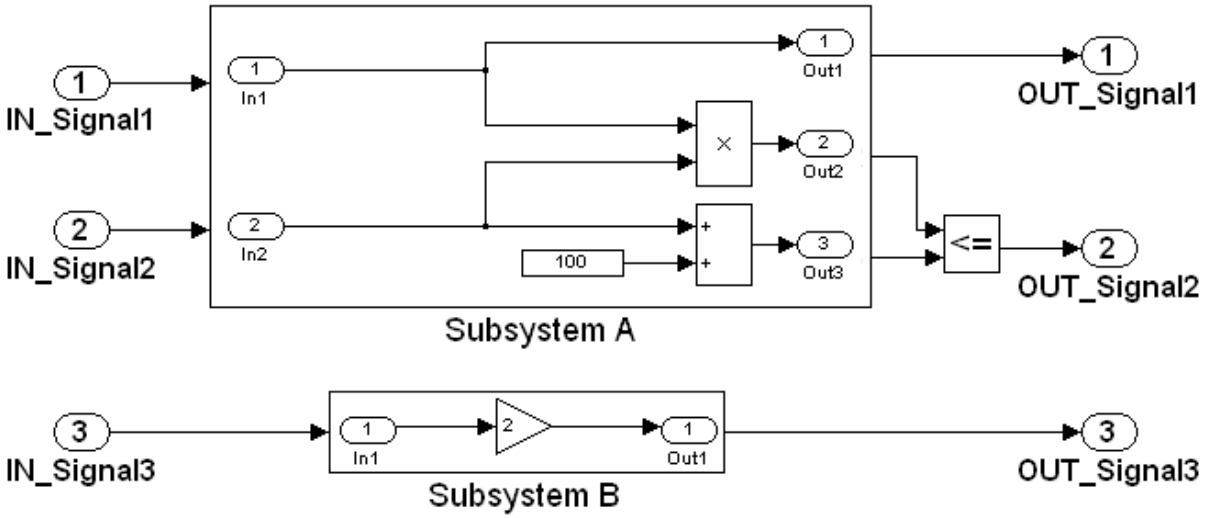


Abbildung 6-6 Simulink-Modellbeispiel - Sprungmarken ersetzt

Im Anschluss wird die Funktion der Simulink-Bibliotheks-Blöcke abstrahiert durch eine Interpretation der möglichen Auswirkung von Eingangssignalen auf Ausgangssignale. Diese Abstraktion ist notwendig, da die Semantik der Simulink-Blöcke nicht grafisch modelliert wird, und auch nicht innerhalb der Modelldateien abgelegt wird. Dabei wird davon ausgegangen, dass jeder Eingangsport eines Simulink-Blocks einen Einfluss auf jeden Ausgangsport des Blocks haben kann. Das Ergebnis dieser Abstraktion ist in Abbildung 6-7 dargestellt.

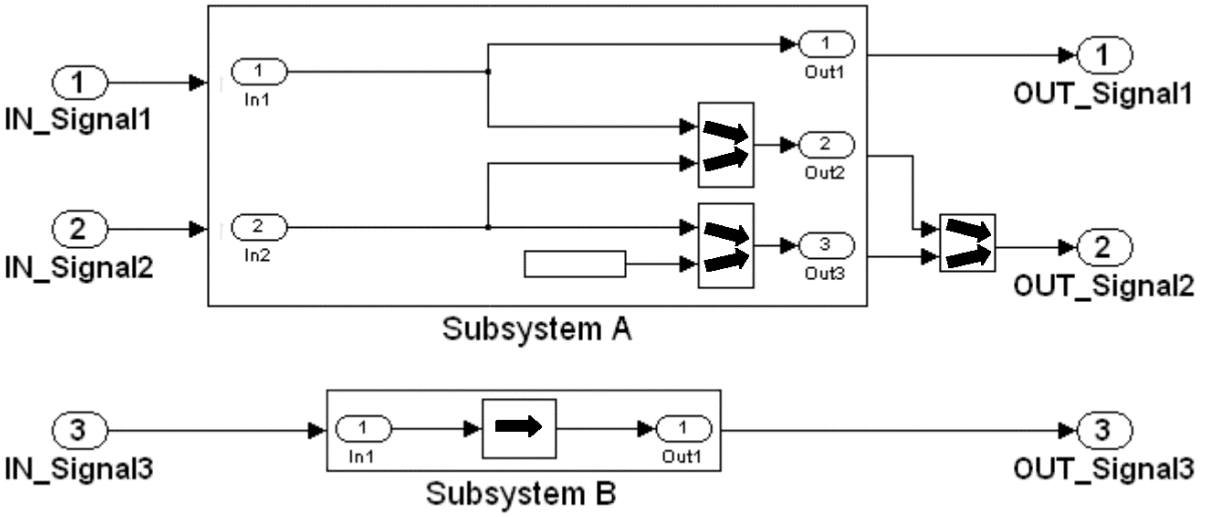


Abbildung 6-7 Simulink-Modellbeispiel – abstrahierte Funktion

Nachdem diese Schritte vollzogen sind, existieren nur noch direkte Komponentenabhängigkeiten in der Analysesoftware. Damit können die Ein- und Ausgangsbeziehungen des Modells anhand einfacher Erreichbarkeitsberechnung ermittelt werden, wie in Abbildung 6-8 verdeutlicht ist. Im Beispiel ist das Ausgangssignal OUT_Signal1 lediglich von IN_Signal1 abhängig. Das Ausgangssignal OUT_Signal3 wird ebenfalls nur von einem Eingangssignal (IN_Signal3) beeinflusst. Dagegen wirken sich zwei Eingangssignale, nämlich IN_Signal1 und IN_Signal2, auf das Ausgangssignal OUT_Signal2 aus.

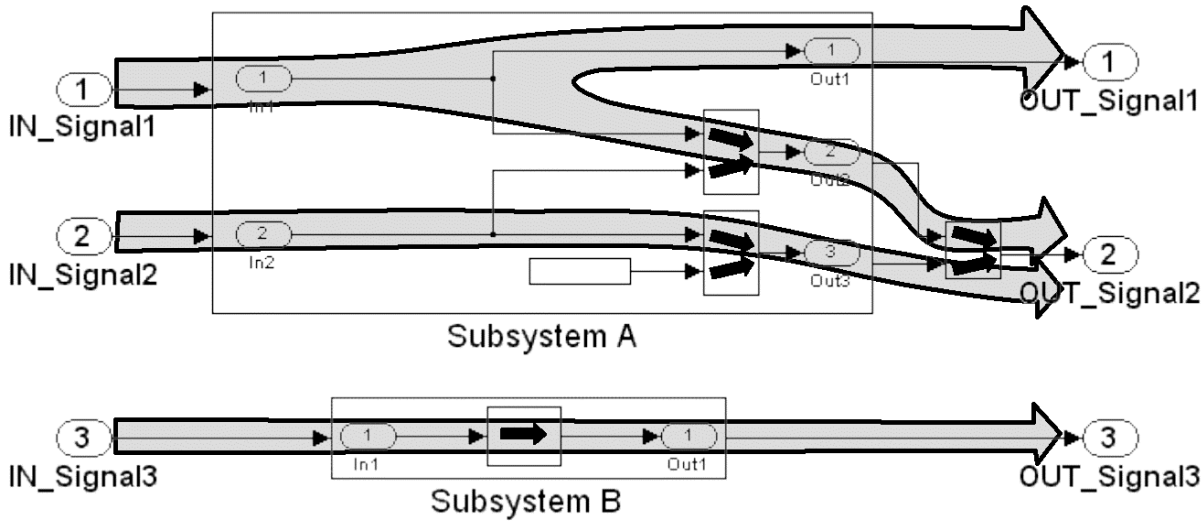


Abbildung 6-8 Simulink-Modellbeispiel - Abhängigkeiten verfolgt

6.5.3.1. Verhalten der Software

Zwischenzeitig gab es Überlegungen, in weiter gehenden Betrachtungen der Strukturanalyse, auch das betriebsbedingte Verhalten der Software in die Abhängigkeitsanalyse einzubeziehen, um am Ende die daraus genauere Aussagen treffen zu können, ob sich das System korrekt verhält.

Dafür sollte für jede Ausgangsgröße des Steuergerätes eine Gleichung angegeben werden, die basierend auf inneren Zuständen des Steuergerätes und seiner Eingangsgrößen die „korrekten“ Ausgangsgrößen ermittelt.

Dieser Ansatz wurde aus mehreren Gründen verworfen:

- Die Koeffizienten der Gleichungen sind nur durch Analyse des Ausgangsprogramms erhältlich, das sich nur in Sonderfällen im Zugriff des OEM befindet.
- Die Koeffizienten der Gleichungen sind zwar prinzipiell automatisiert aus dem Quellcode ableitbar, allerdings würde die Ausführung einer solchen Gleichung gleichbedeutend mit

dem parallelen Abarbeiten eines dem Ursprungsquellcode äquivalenten Programms sein. Bei korrekter Umsetzung darf daher niemals ein Fehler gefunden werden.

- Das Diagnosesystem betrachtet auch innere Zustände, wie z.B. Wartezeiten oder Adaptionswerte. Daher muss es auch dieselben Eingaben des Steuergerätes zum jeweils gleichen Zeitpunkt wie das ursprüngliche Programm erhalten. Nur so kann die Konsistenz der beiden Ergebnisse gewahrt bleiben. Jede Unterabtastung durch die Diagnosesoftware kann damit zu Fehlalarmen der Diagnose führen. Eine Speisung der Diagnosefunktionen mit der gleichen Abtastrate wie die Nennfunktionalität erfordert eine hohe Rechenleistung.
- Eine semantische Vereinfachung zur Beschleunigung der Berechnung der Zustands-Gleichungen für das Diagnosesystem ist ebenfalls aus Konsistenzgründen nicht möglich. Auch geringe Änderungen an der Bedeutung einer Gleichung führen dazu, dass der Vergleich mit der Ursprungs-Funktion an verschiedenen Berechnungspunkten in Unterschieden resultiert. Die für die Diagnose relevante Unterscheidung zwischen Fehlerfall und Vereinfachungs-Abweichung ist nicht automatisiert zu treffen.
- Aus demselben Grund kann auch kein Grobentwurfsmodell zum Überprüfen des Steuergerätes während des Betriebes genutzt werden.
- Aus ebenfalls demselben Grund ist eine Abbildung des Modells in eine Gleichung praktisch unmöglich, denn für eine korrekte Abbildung muss das logische Verhalten jedes eingesetzten Modell-Bibliotheksblocks sowie die Abarbeitungsreihenfolge der Modellblöcke bekannt sein. Dies ist für viele Modellkomponenten nicht exakt gegeben.

6.5.4. Kombination der Strukturmodelle

In einem folgenden Schritt müssen für die systemweite Diagnose die entstandenen Teilmodelle zu einem System zusammen gestellt werden.

Durch die vorhergehenden Schritte der Abstraktion der verschiedenen Teilmodelle auf bloße Abhängigkeiten von Strukturkomponenten untereinander sind die Teilmodelle trotz ihrer unterschiedlichen physikalischen bzw. logischen Bedeutung uneingeschränkt integrierbar.

Dazu sind prinzipiell zwei Wege denkbar. Zum Einen kann die Nomenklatur der Strukturelemente so gewählt werden, dass die drei unabhängigen Teilstrukturen gleiche Schnittstellenbezeichner haben, und somit anhand ihres Namens direkt zusammen gefügt werden können. Zum Anderen können Tabellen erstellt werden, die übereinstimmende Strukturelemente der Teilmodelle identifizieren und so die Schnittstelle zwischen Teilmodellen bilden. Durch den

Einsatz solcher Übersetzungstabellen kann die Systemstruktur automatisiert zusammen gesetzt werden. Für das vorliegende Anwendungsbeispiel wurden entsprechende Tabellen in DOORS im Modul „Hardware- und Software Systemkopplung“ vorgeschlagen und integriert.

6.6. Zuordnung Fehlereinträge zu Strukturelementen

Es gibt prinzipiell zwei unterschiedliche erzeugte Typen von Fehlercodes, die in einem Steuergerät abgelegt werden können. Das sind zum Einen diejenigen, die direkt Eingangssignale oder Aktoren überwachen, und dabei Verletzungen der Signalbeschreibung detektieren. Diese Art Fehlercode lässt sich direkt einem Strukturelement zuordnen. Dazu wird im Regelfall diejenige Hardwarekomponente belastet, die das fehlerhafte Signal detektiert hat. Die Fehlerursache ist in der Abhängigkeitsstruktur vor der Signalsenke, nämlich dem belasteten Strukturelement zu suchen. Zum Anderen gibt es Fehlercodes, die aufgrund der Kombination bestimmter Signal- oder Modellzustände auftreten können oder vom Gesetzgeber vorgeschrieben werden. Das sind zum Beispiel Verbrennungsunregelmäßigkeiten an Motoren oder ausgebliebene Reaktionen von Steuergliedern. In diesem Fall müssen die zu belastenden Strukturelemente zu dem Fehlercode z.B. in einer FMEA herausgefunden und explizit angegeben werden.

Generell erfolgt die Zuordnung von einem Fehlercode für beide Varianten von Fehlercodetypen in der Spezifikation unter der Überschrift „Fehlercodes“ anhand einer festgelegten Notation, damit die Zuordnung automatisch ausgewertet werden kann.

6.7. Prüfung des Diagnosesystems

Die ausgearbeiteten Diagnosefunktionen und Strukturanalysen müssen bei Abschluss der Entwicklung validiert werden. Da das Diagnoseverfahren selbst unabhängig von den Diagnosemodellen ist, braucht dieses nur bei der Erstumsetzung ausführlich geprüft zu werden.

In späteren Anwendungen können dann lediglich Fehler in den Eingangsgrößen für das Diagnoseverfahren wie Fehlersetzbedingungen und bei der Schnittstellenbeschreibung zwischen Hardware und Software sowie bei der Beschreibung der Beziehung von Fehlercodes zu belasteten Komponenten auftreten.

Wesentliche Voraussetzung für das Funktionieren der strukturbasierten Diagnose ist eine belastbare Basis an Fehlererkennungsmechanismen der Steuergeräte selbst. Hier ist daher eine verlässliche Fehlercodegenerierung notwendig. Zu diesem Zweck müssen in der Spezifikation für alle Signale, die ein Steuergerät verarbeitet, eindeutig definierte Wertebereiche und ein-

deutig bestimmte Fehlersetzbedingungen angegeben sein. Mit diesen Vorgaben lässt sich nun die Korrektheit der Steuergeräteeigendiagnose bezüglich der Fehlercodesetzung und damit der Eingangsdaten für das Systemdiagnoseverfahren überprüfen.

Die Schnittstellenbeschreibung der Kopplung zwischen Hardware und Software wird durch eine Konsistenzprüfung erreicht, bei der alle Softwareschnittstellen eine Verbindung zu einer Hardwarekomponente aufweisen müssen. Dass dabei alle diese Beziehungen korrekt hergestellt wurden, lässt sich nicht durch das Diagnosesystem feststellen, sondern nur bei der Erstellung des zugehörigen Spezifikationabschnittes durch Sorgfalt des zuständigen Entwicklers sicherstellen.

Die korrekte Zuordnung von Fehlercodes zu Systemkomponenten lässt sich ebenfalls nur durch eine Konsistenzprüfung über das Vorhandensein der belasteten Systemkomponente überprüfen.

Für einen Test des spezifischen Diagnosesystems für eine Fahrzeugentwicklung, für das beim Vorliegen bestimmter Fehler eine bestimmte Reaktion des Diagnosesystems erwartet wird, müssen alle Eintrittsbedingungen in einer Erweiterung der Spezifikation explizit notiert werden. Diese Fälle sind als Testfälle für die Diagnose anzusehen, und prüfen das Zusammenspiel der zusammengesetzten Strukturdaten für das Diagnosesystem.

7. Fallstudie Diagnose Kraftfahrzeug-Komfortsystem

Die vorgestellten Diagnosemöglichkeiten wurden im Rahmen des Projektes an mehreren verschiedenen Demonstrationsobjekten gezeigt und auf ihre Praxistauglichkeit hin überprüft.

Dabei ging es zum Einen um die Bestätigung erster Umsetzungen zur Diagnose anhand von Hardwarestrukturen. Zum Anderen sollte gezeigt werden, dass die dabei erzielten Erkenntnisse ohne Bruch der Diagnosemethode direkt um die Software- und Kommunikationsabhängigkeiten ergänzt werden können.

Die eigentliche Validierungsphase wurde an dem für STEP-X entwickelten Komfortsystemdemonstrator durchgeführt. In das Demonstrationsobjekt sind vier innerhalb des STEP-X-Projektes entwickelte Türsteuergeräte installiert, mit denen die Fenster- und Spiegelfunktionen des Komfortsystems eines Volkswagen Polo anhand der Vorgaben aus der Spezifikation durchgeführt werden können.

Für die Unterstützung der vorgestellten Diagnose-Funktionen wurde die Hardwarestruktur in die Spezifikation einbezogen. Ebenso wurde die STEP-X-Eigenentwicklung der Software und der Kommunikation auf Abhängigkeiten untersucht und in das Strukturmodell integriert. In diesem Abschnitt wird konkret anhand des zur Validierung herangezogenen Beispiels die Umsetzung beschrieben.

In einer anschließenden Untersuchung wurde gezeigt, dass die Softwarestrukturanalyse neben den Beispiel-Modellen aus STEP-X auch ein zur Verfügung stehendes seriennahes Modell eines Bordnetzsteuergerätes analysieren kann.

7.1. Kopplung der Umgebungsmodelle mit der Spezifikation

Die Schnittstellen eines Steuergerätes mit der Außenwelt müssen zunächst klar definiert in der Spezifikation beschrieben sein. Dazu gehört einerseits die Definition relevanter physikalischer Parameter, d.h. geforderte Leitungsquerschnitte, Kabelfarben, Isolierungsmaterial für Kabelstränge etc. Diese zusätzlichen Informationen sollen im Folgenden nicht näher betrachtet werden, da sie aus diagnostischer Sicht nicht relevant sind. Daher wird für jedes vom Steuergerät ausgewertete Signal in der Spezifikation eine Struktur wie in Abbildung 7-1 gezeigt gefordert.

Anhand der Methodikvorgaben aus [Mutz05] werden die Systemschnittstellen des zu spezifizierenden Steuergerätes zunächst aufgeteilt in die Bereiche Bedienschnittstellentypen, Aktortypen und Sensortypen.

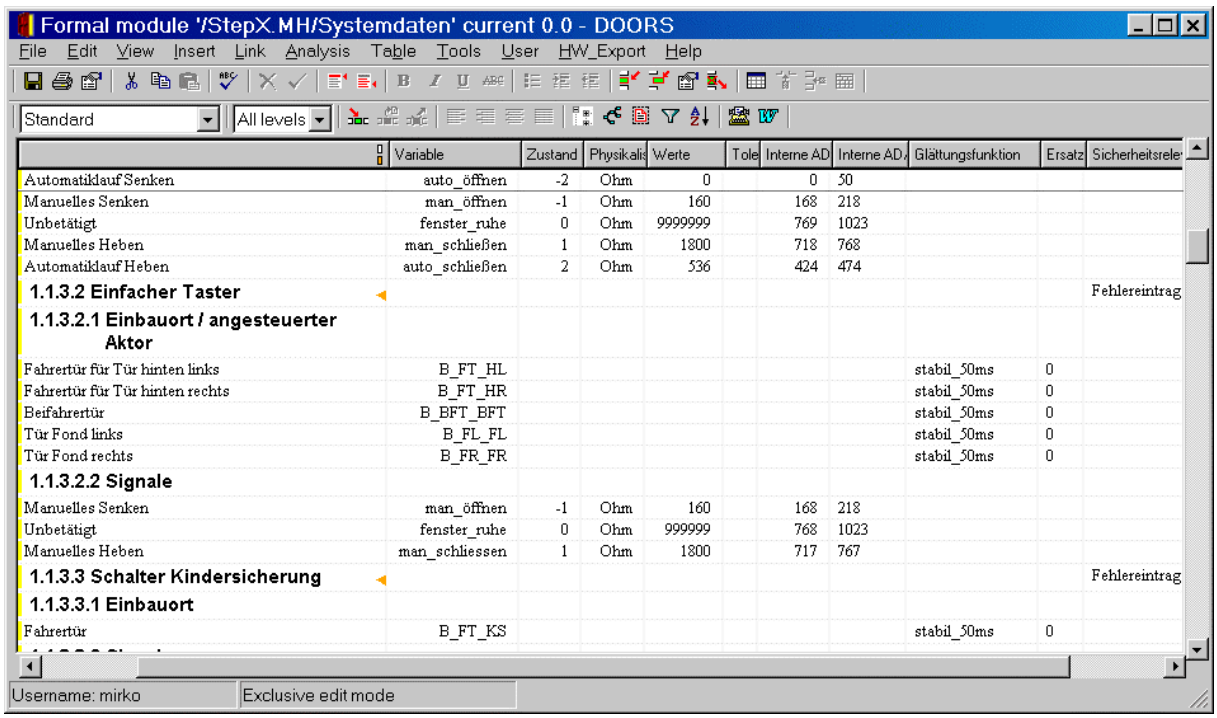


Abbildung 7-1 Definition der Hardwareschnittstellen aus Softwaresicht

Zu allen Elementtypen werden dann die im Steuergerät tatsächlich vorhanden Elemente mit dem zugehörigen Einbauort aufgelistet. Um die Relevanz des Elementes für das System markieren zu können, wird die notwendige Sicherheitsrelevanz bei einem erkannten Ausfall notiert. Die zulässigen Sicherheitsrelevanzgrade sind in Tabelle 7-1 dargestellt.

Schweregrad	Spezifikationseintrag	Beschreibung
1	Fehlereintrag	Nur zur Dokumentation für die Werkstatt bei der nächsten Routineinspektion. Es handelt sich um einen Fehler, der ausschließlich von dem eintragenden Steuergerät detektiert werden kann, und die Betriebssicherheit des Kraftfahrzeugs wird nicht beeinträchtigt.
2	Systemfehlereintrag	Nur zur Dokumentation für die Werkstatt bei der nächsten Routineinspektion. Der Fehler könnte als Ursache einen Systemfehler haben. Der Fehler beeinträchtigt die Betriebssicherheit des Kraftfahrzeugs nicht.
3	Schwerer Fehler	Der erkannte Fehler führt zu einer wesentlichen Beeinträchtigung des Kraftfahrzeugs oder der Umgebung in Form von z.B. erhöhtem Verbrauch oder erhöhtem Abgasausstoß. Der Fahrer soll aufgefordert werden, eine Werkstatt aufzusuchen, um den Fehler beheben zu lassen.
4	Kritischer Fehler	Der erkannte Fehler gefährdet die Fahrzeuginsassen und andere Verkehrsteilnehmer. Aufforderung an den Fahrer, das Fahrzeug sofort stillzulegen.

Tabelle 7-1 Relevanz einer erkannten Fehlfunktion

Die vorzusehende Filterung der analogen Eingangssignale durch das Steuergerät werden an gleicher Stelle in der Spezifikation niedergelegt. Nur so lassen sich die Nenn- und Diagnosefunktionen im Rahmen der Tests definiert überprüfen. Beispiele für mögliche Filterfunktionen, wie sie in der Spezifikation gefordert sein können und im Anwendungsbeispiel genutzt werden, sind in Tabelle 7-2 beschrieben.

Spezifikationsbezeichnung	Parameter	Beschreibung
Stabil	Dauer	Übernimmt einen Eingangswert erst, wenn er die vorgegebene Dauer kontinuierlich angelegen hat.
Median	Anzahl der Werte, Abtastrate	Bestimmt den Median des rauschenden Eingangssignals aus einem bestimmten Satz von Messwerten.
Butterworth-Tiefpass	Ordnung, Grenzfrequenz	Anhand eines Butterworthfilters verarbeitetes Eingangssignal.
Tschebyscheff-Tiefpass	Ordnung, Grenzfrequenz	Anhand eines Tschebyschefffilters verarbeitetes Eingangssignal.
Besseltiefpass	Ordnung, Grenzfrequenz	Anhand eines Besseltiefpasses verarbeitetes Eingangsignal.

Tabelle 7-2 Filterfunktionen für Analogeingänge

Im Eintrag „Einbauort / angesteuerter Aktor“ wird für die Softwaresteuerung zusätzlich der für das spezielle Element zu verwendende Variablenbezeichner eingeführt (Spalte Variable). In Abbildung 7-1 ist das z.B. für den Fahrertaster für das linke Fondfenster die Bezeichnung B_FT_HL.

Zu jedem Signal muss beim erkannten Auftreten von Fehlern eine angemessene Reaktion definiert werden, für Sensoren heißt das, dass in der Spezifikation festgeschrieben wird, wie der Ersatzwert beim Vorliegen eines Defektes lautet. Für den Ausfall eines Sensors funktioniert damit die entwickelte Software immer noch in einem überprüfbaren und definierten Rahmen.

Für jedes Signal muss eindeutig die physikalische Einheit der gemessenen Größe definiert werden, um dies notfalls in der Werkstatt überprüfen zu können. Im Beispiel ist als physikalische Einheit für die meisten Einträge „Ohm“ vorgesehen, da vornehmlich Tasterstellungen beschrieben werden und die Tasterstellungen bei Volkswagen durch unterschiedliche Widerstandswerte kodiert werden. Für diese Typen von Sensoren mit diskreten Tasterstellungen, die analog eingelesen werden, sind weitere Angaben in der Spezifikation notwendig. Durch die Angabe von Gültigkeitsintervallen kann das analoge Signal auf diskrete Zustände abgebildet werden. Die daraus ermittelten diskreten Zustände können von dem Funktionsentwickler ent-

weder über den in der Spezifikation beschriebenen symbolischen Bezeichner oder über den ebenfalls dort notierten Zahlenwert des Zustands verwendet werden.

7.2. Strukturanalyse der Steuerungsmodelle

Die Steuerungsmodelle innerhalb der Modellierungswerkzeuge haben nur geringe zusätzliche Anforderungen zu erfüllen, um durch die beschriebenen Verfahren automatisiert nach diagnoserelevanten Abhängigkeiten untersucht werden zu können. Am folgenden Beispiel ist anhand der Abbildung 7-2 dargestellt, wie sich die Funktionsabhängigkeiten eines seriennahen Simulink-Steuergerätemodells darstellen. Dazu wurden zunächst alle Hierarchieebenen des Modells aufgelöst und alle Referenzzugriffe über symbolische Namen und gebündelte Signale ersetzt, wobei jedes einzelne Signal von seinem Ursprung zu seiner Senke verfolgt wird. Eine Signalquelle bzw. eine Signalsenke bezeichnet hier einen Block, der nicht weiter aufgelöst werden kann. Dazu zählen beispielsweise Bibliotheksfunktionen von Simulink sowie Ein- und Ausgabeschnittstellen. Eine Analyse dieser Signalabhängigkeiten resultiert in einer komplexen Abhängigkeitsstruktur wie in Abbildung 7-2 dargestellt.

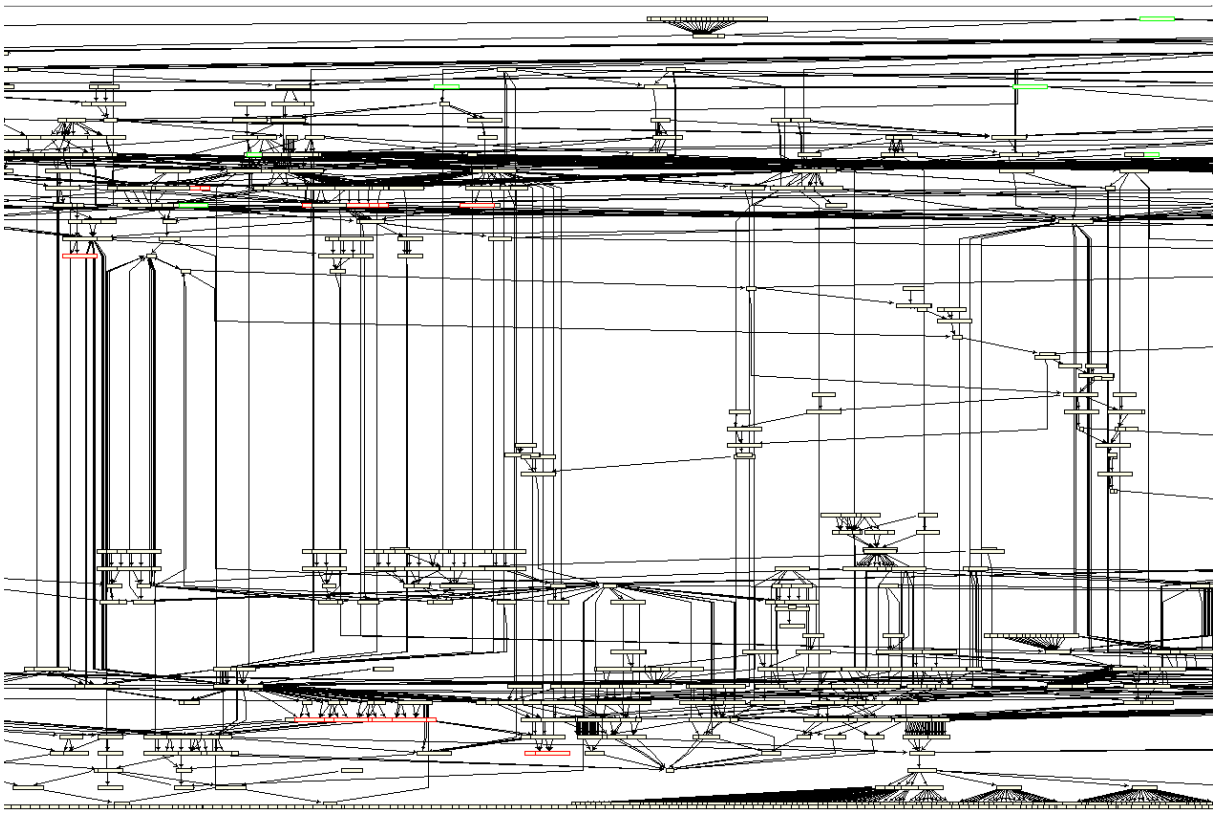


Abbildung 7-2 Auszug der Abhängigkeiten eines Simulink-Modells vor der Analyse

Für die Diagnose ist die Ablage eines so detaillierten Abhängigkeitsmodells allenfalls in der Entwicklungsphase notwendig. In späteren Entwicklungsphasen, wie in der Serienfertigung, sind lediglich die Beziehungen zwischen Eingangsschnittstellen des Steuergerätes und seinen Ausgangsschnittstellen relevant. Diese Reduktion des Ursprungsmodells auf Ein- und Ausgangsbeziehungen stellt Abbildung 7-3 dar. Im oberen Bildbereich sind die Eingänge des Systems zu sehen, während im unteren Bildbereich die Ausgangsschnittstellen dargestellt sind. Die Abhängigkeit von bestimmten Eingangsschnittstellen ist durch die Pfeile angedeutet.

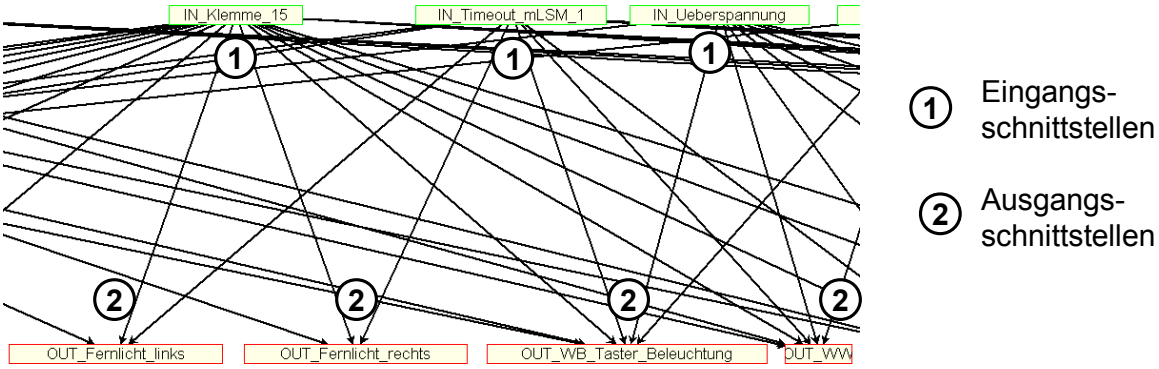


Abbildung 7-3 Auszug eines Simulink-Modells reduziert auf Ein-Ausgangsbeziehungen

7.3. Kopplung zur Spezifikation

Die für die Diagnose nutzbaren bzw. auszuwertenden elektrischen Daten sind in der Spezifikation integriert. Dafür ist für jede Entwicklung ein Modul obligatorisch, das die Schwerpunkte Hardwarestruktur, Softwareschnittstellen und die gültigen Fehlercodes behandelt. Dabei bildet die Hardwarestruktur in automatisiert ausgewerteter Form die Elektrik des Bordnetzes ab. Die Softwareschnittstellen kennzeichnen den Übergang zwischen Hardware und Software. Um die einzelnen Elemente mit Fehlern belasten zu können, werden die von der Diagnose auszuwertenden Fehlercodes zusammen mit den jeweils belasteten Komponenten ebenfalls in dieses Modul eingetragen. Das genaue Vorgehen für die automatisierbare Verarbeitung dieser Informationen wird in den folgenden Abschnitten erläutert.

7.3.1. Hardwarestruktur

Die Hardwarestruktur des zu entwerfenden Systems wird ebenfalls in der Spezifikation beschrieben und dient sowohl als Vorgabe für Zulieferer, den Kabelbaum entsprechend zu gestalten, wie auch zur Dokumentation des elektrischen Systems. Im Projekt wurde die in Abbildung 7-4 dargestellte Notation verwendet. Diese bietet über die Skriptsprache DXL von DOORS die Möglichkeit der automatisierten Auslesung.

Die Dokumentstruktur der Hardwarebeschreibung ist grundsätzlich aufgeteilt in drei große Blöcke. Im ersten Teil wird die Hardware textuell grob beschrieben. Dieser Teil dient nur der Übersicht und wird nicht automatisch ausgewertet. Im zweiten Teil ist der elektrische Schaltplan des zu entwickelnden Systems enthalten. Dieser muss für die automatische Auswertung mit „Hardwarestruktur“ bezeichnet sein.

In diesem Block werden alle Komponenten des elektrischen Systems als ein eigenes Unterkapitel der Hardwarestruktur angelegt. Zu jedem dieser Blöcke werden alle zugehörigen Pins mit symbolischen Signalnamen verbunden. Im Beispiel der Abbildung 7-4 ist der Pin mit der Bezeichnung ANA1_1 mit dem Signal mit dem symbolischen Namen BF_FT_FT verbunden. Als zusätzliche Information wird in der Spezifikation in der Spalte „Variante“ notiert, bei welchen Varianten diese Beschaltung genutzt werden soll.

Die Spalte „extern“ kennzeichnet für jedes Signal einzeln, ob es in dem Schaltplan generiert wird, der durch das DOORS-Dokument aufgespannt wird. Kommen elektrische Verbindungen aus anderen Systemen, so wird das Signale als extern markiert.

Für die Verfolgung der Auswirkung von Fehlern im System ist die Einordnung der elektrischen Signale in Signalquellen und Signalsenken notwendig. In der Beispielabbildung sind die analogen Eingänge des STEP-X-Demonstrators gezeigt, diese sind durch die Kennung „In“ in der Spalte „Quelle/Senke“ als Signalsenken bezeichnet.

	Port	Variante	extern	Quelle/Senke
Beschaltung des STEP-X Demonstrators	n/a	Allg	False	keine
1.2 Kontext	n/a	Allg	False	keine
Die Schaltung beinhaltet das Komfortsystem eines PQ24 bestehend aus den 4 Türsteuergeräten mit ihrer angeschlossenen Hardware.	n/a	Allg	False	keine
2 Hardwarestruktur	n/a	Allg	False	keine
2.1 SG Fahrer	n/a	Allg	False	keine
BF_FT_FT	ANA1_1	Allg	False	In
BF_FT_BFT	ANA1_2	Allg	False	In
BF_FT_HL	ANA1_3	Allg	False	In
BF_FT_HR	ANA1_4	Allg	False	In
BF_FT_KS	ANA1_5	Allg	False	In
BF_FT_ZVS	ANA1_6	Allg	False	In

Abbildung 7-4 Definition der Hardwarestruktur in DOORS

An Steckern muss die elektrische Verfolgung der Signale über die Steckverbindung hinaus möglich sein. Um jedoch die Richtung des Signals verfolgen zu können, muss zu jedem Pin ein linkseitiger Signalname und ein rechtsseitiger Signalname angegeben werden. In

Abbildung 7-5 wird beispielsweise das Signal „BF_FT_HOCH“ an Pin ANA1_4 des Steckers ANA1 am Beifahrersteuergerät mit dem Signal „Leitung_BF_BFT_HOCH“ verbunden. Dieses führt zum Bedienfeld für den Beifahrerfensterheber.

ID		Port	Variante	extern	Quelle/Senke
669	**2.16 Stecker ANA1 SG Beifahrer**	n/a	Allg	False	keine
952	Leitung_BF_BFT_HOCH	ANA1_4	Allg	False	In
953	Leitung_BF_BFT_RUNTER	ANA1_5	Allg	False	In
954	Leitung_BFT_58D_HS	ANA1_7	Allg	False	Out
955	Leitung_BF_BFT_GND	ANA1_8	Allg	False	Out
674	BF_BFT_HOCH	ANA1_4	Allg	False	Out
675	BF_BFT_RUNTER	ANA1_5	Allg	False	Out
677	BFT_58D_HS	ANA1_7	Allg	False	In
678	BF_BFT_GND	ANA1_8	Allg	False	In
679	**2.17 Stecker Bedienfeld FH Beifahrer**	n/a	Allg	False	keine
956	Leitung_BF_BFT_HOCH	A1	Allg	False	Out
957	Leitung_BF_BFT_RUNTER	A2	Allg	False	Out
958	Leitung_BFT_58D_HS	A4	Allg	False	In
959	Leitung_BF_BFT_GND	A3	Allg	False	In
681	BF_BFT_HOCH-1	A1	Allg	False	In
682	BF_BFT_RUNTER-1	A2	Allg	False	In
687	BFT_58D_HS-1	A4	Allg	False	Out
688	BF_BFT_GND-1	A3	Allg	False	Out
689	**2.18 Bedienfeld FH Beifahrer**	n/a	Allg	False	keine
690	BF_BFT_HOCH-1	A1	Allg	False	Out
691	BF_BFT_RUNTER-1	A2	Allg	False	Out
696	BFT_58D_HS-1	A4	Allg	False	In
697	BF_BFT_GND-1	A3	Allg	False	In

Abbildung 7-5 Definition der Weiterleitung von Steckersignalen

7.3.2. Softwareschnittstellen

Um die Abhängigkeiten zwischen Software und Hardware eindeutig bestimmen zu können, sind die Schnittstellen zwischen Hardware und Software klar zu dokumentieren. Die dafür notwendige Notation innerhalb der DOORS-Module der Spezifikation ist angelehnt an die Hardwarestrukturdefinition.

Wichtig ist in diesem Zusammenhang die Überschrift „Software“, damit die Auswertungssoftware automatisch die zugehörigen Verbindungen erstellen kann. Die Nomenklatur sieht vor, dass die Signalbezeichnungen anhand der hierarchischen Softwarestrukturdefinition aus Simulink gewählt werden. Als Port wird die der Softwareschnittstelle entsprechende Hardwareschnittstelle angegeben. Die Hardwareschnittstellenbezeichnung wird nach dem Schlüssel „Komponente\Pin“ notiert. Die Richtung des Informationsflusses wird wie bei der Hardwaredefinition in der Spalte „Quelle/Senke“ festgelegt.

Im Beispiel der Abbildung 7-6 ist im Simulink-Modell „ecu_fahrer“ das Modellelement „IN_BS1_FhzGeschw“ mit dem Pin „CAN3“ des Steuergerätes Fahrer („SG Fahrer“) verbunden. „IN_BS1_FhzGeschw“ ist ein Eingangssignal, das von „CAN3“ seine Informationen erhält.

ID		Port	Variante	extern	Quelle/Senke
1041	**2.44 Software**	n/a	Allg	False	keine
1070	"ecu_fahrer"\"IN_BS1_FhzGeschw"	SG Fahrer\CAN3	Allg	False	In
1071	"ecu_fahrer"\"IN_BS1_FH_Freigabe"	SG Fahrer\CAN3	Allg	False	In
1072	"ecu_fahrer"\"IN_ZA1_ZAS_K1_15"	SG Fahrer\CAN3	Allg	False	In
1073	"ecu_fahrer"\"IN_Sleep_Ack"	SG Fahrer\CAN3	Allg	False	In
1126	"ecu_fahrer"\"IN_fenster_pos"	SG Fahrer\PWR1_14	Allg	False	In
1127	"ecu_fahrer"\"IN_fenster_pos"	SG Fahrer\PWR1_15	Allg	False	In
1076	"ecu_fahrer"\"IN_imotf"	n/a	Allg	False	In
1077	"ecu_fahrer"\"IN_bf_ft"	SG Fahrer\ANA1_1	Allg	False	In
1042	"ecu_fahrer"\"IN_bf_bft"	SG Fahrer\ANA1_2	Allg	False	In

Abbildung 7-6 Definition der Softwareschnittstellen in DOORS

7.3.3. Definition von Fehlercodes

Durch die Einführung von Fehlercodes können Strukturelemente als verdächtige Komponenten belastet werden. Durch die Kenntnis der Gesamtstruktur und die darin manifestierten Abhängigkeiten der Strukturelemente voneinander kann der Fehlereinfluss verfolgt werden.

Dazu müssen für jeden Fehlercode diejenigen Komponenten in der Spezifikation dokumentiert werden, die nach Ermessen des Spezifikationserstellers ursächlich für sein Auftreten sein können. Die Definition von Belastungen durch Fehlercodes in der Spezifikation orientiert sich ebenfalls anhand der Hardwarestrukturbeschreibungsnotation und wird im selben Modul beschrieben. Die von der Auswertungssoftware erwartete Überschrift für Fehlercodes lautet „Fehlercodes“. Die weitere hierarchische Gliederung der Überschriften dient nur der Übersichtlichkeit der Spezifikation, hat aber keinen Einfluss auf die Diagnoseeigenschaften.

Zur Definition eines zu belastenden Strukturelementes wird in jeder Zeile als erster Eintrag die hexadezimal notierte Fehlercodenummer erwartet. Evtl. vorhandene textuelle Beschreibungen des Fehlercodes können an die Fehlercodenummer angehängt werden. In der Spalte „Port“ wird diejenige Hardwarekomponente notiert, mit der ein Steuergerät den Fehler feststellt. Im Beispiel der Abbildung 7-7 belastet ein unplausibles Signal der X/Y-Spiegelverstellung des Fahrers den Analogeingang „SG Fahrer\ANA2_1“. Fehlerursache können alle Komponenten sein, von denen dieses Element abhängig ist. Die Spalten „extern“ und „Quelle/Senke“ haben für die Fehlercodeauswertung keine Bedeutung.

ID		Port	Variante	extern	Quelle/Senke
1197	**3 Fehlercodes**	n/a	Allg	False	keine
1198	**3.1 SG Fahrer**	n/a	Allg	False	keine
1203	0001 - VL Signal FH VL	SG Fahrer\ANA1_1	Allg	False	keine
1206	0002 - VL Signal FH VR	SG Fahrer\ANA1_2	Allg	False	keine
1208	0003 - VL Signal FH HL	SG Fahrer\ANA1_3	Allg	False	keine
1209	0004 - VL Signal FH HR	SG Fahrer\ANA1_4	Allg	False	keine
1207	0005 - VL Signal KISI	SG Fahrer\ANA1_5	Allg	False	keine
1211	0006 - VL Signal ZVS	SG Fahrer\ANA1_6	Allg	False	keine
1210	0010 - VL Signal Spiegel X/Y	SG Fahrer\ANA2_1	Allg	False	keine
1213	0011 - VL Signal Spiegel R/L	SG Fahrer\ANA2_6	Allg	False	keine

Abbildung 7-7 Definition der Fehlercodes in DOORS

7.4. Anwendung am STEP-X Prüfstand

Als primäres Anwendungsbeispiel für die Untersuchungen dient ein aus Originalteilen nachgebildetes Komfortsystem eines Volkswagen Polo in einem Komfortsystemprüfstand der in Abbildung 7-8 dargestellt ist.

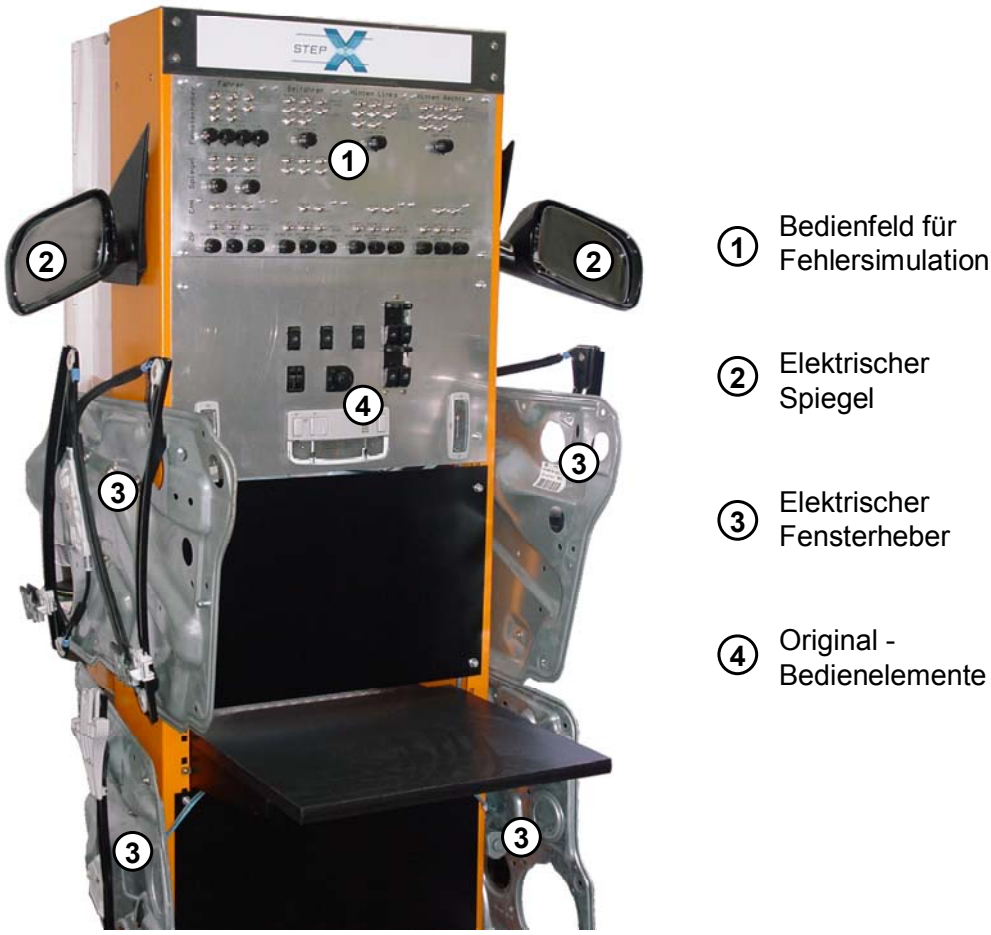


Abbildung 7-8 STEP-X Komfortsystemprüfstand

Das betrachtete Komfortsystem besteht aus der elektrischen Fensterheber- und Spiegelsteuerung sowie den Sensoren, Aktoren und Bedieneinrichtungen eines Volkswagen Polo. Im Rahmen der Arbeiten von STEP-X wurden hieran die Durchgängigkeit der Werkzeugkette vom Entwurf bis zum Seriencode gezeigt. Das System besteht aus vier prototypischen Entwicklungssteuergeräten, die über CAN-Bus miteinander vernetzt sind und die Sensorik und Aktorik bedienen können.

Für die Diagnose ist im Kopfbereich des Prüfstandes ein Panel eingelassen, das elektrische Fehler im System simulieren kann. Die Palette an Fehlern reicht hier von elektrischen Kurzschlüssen zur Betriebsspannung, zwischen Leitungen und zur Masse, über Alterung von Tasterbaugruppen bis hin zu Kommunikationsstörungen.

Für tiefer gehende Prüfungen der Testfunktionen entstand im Rahmen von STEP-X zusätzlich der Einzeltür-Prüfstand aus Abbildung 7-9[Hors05]. Hier können alle elektrischen Funktionen einer Beifahrertür mittels einer Testinstrumentierung automatisch angesteuert werden. Die entwickelten Funktionsmodelle können hier wahlweise auf einem Steuergeräte-Prototypen oder innerhalb einer PC-Simulation ablaufen.

Abbildung 7-9 STEP-X Versuchsträger

Die entwickelten Modelle und Diagnosefunktionen konnten zum Abschluss auch an dem STEP-X Versuchsfahrzeug überprüft werden, das ebenfalls in Abbildung 7-9 dargestellt ist. Hierzu wurden die vier Originalsteuergeräte des Komfortsystems aus dem Polo entfernt, und durch die Steuergeräte-Prototypen ersetzt.

7.4.1. Eingesetzte Software

Für die Realisierung des vorgeschlagenen Diagnoseverfahrens wurden unterschiedliche Entwicklungswerkzeuge eingesetzt. Für die Entwicklung der Strukturanalysesoftware auf PC-Basis wurden die Algorithmen in der Programmiersprache C in der Visual Studio 6.0 Entwicklungsumgebung von Microsoft entwickelt.

Die Implementierung der strukturbasierten Algorithmen auf dem Mikrocontroller wurden ebenfalls in C aber innerhalb der Entwicklungsumgebung für Eingebettete Systeme Tasking EDE für C167 Mikrocontroller durchgeführt.

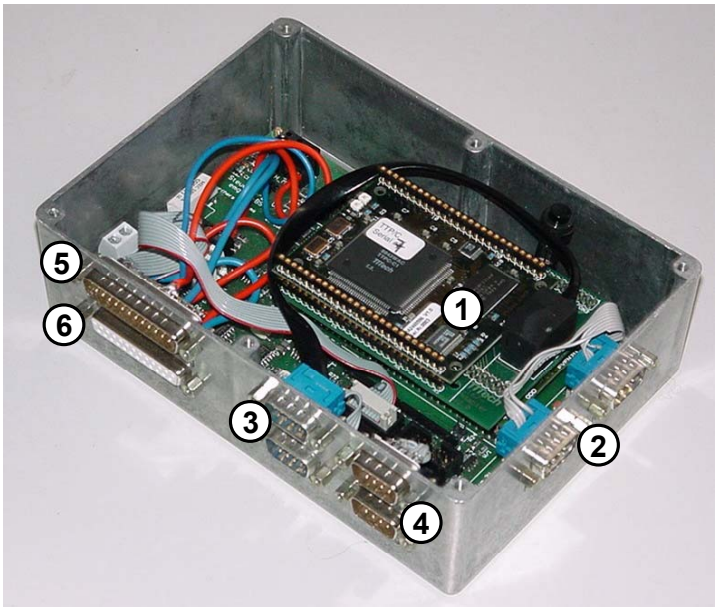
In DOORS 7.0 von Telelogic wurden die notwendigen Erweiterungen an der Spezifikation für die Beschreibung der Hardwareschnittstellen und Netztopologien realisiert. Die automatische Auswertung der abgelegten Inhalte wurde mit Hilfe der speziell für DOORS verfügbaren Skriptsprache DOORS Extension Language DXL entwickelt.

Die Nennfunktionalität der Steuergeräte ist aus dem STEP-X Prozess innerhalb von Simulink von The Mathworks entstanden. Die Schnittstellenfunktionalitäten der Software wurden anhand der Spezifikationsvorgaben in DOORS realisiert.

Die implementierte Steuergerätefunktionalität aus Simulink wurde über den von The Mathworks vertriebenen Codegenerator Embedded Coder in C-Code gewandelt und zusammen mit den in C implementierten Hardwaretreiberbibliotheken übersetzt und in ausführbaren C167-Code gebunden.

7.4.2. Eingesetzte Hardware

Für alle Demonstrationssysteme wurden im Rahmen dieser Arbeit die in Abbildung 7-10 dargestellten Steuergeräte entwickelt und eingesetzt. Sie basieren auf Minimodulen der Firma Phytec ausgestattet mit C167 Mikrocontrollern. Die verwendeten Steuergeräte und die darauf vorhandene Sensorik und Aktorik soll im Folgenden näher beschrieben werden.



1. Phytec MiniMODUL C167 (mit TTP/C Modul)
2. 2x Serielle Schnittstelle
3. CAN und TTP/C
4. Analog-Inputs
5. Fensterheber-Steuerung
6. Spannungsversorgung und Spiegelverstellung

Abbildung 7-10 Verwendeter Steuergeräteprototyp

Die Prototyping-Hardware der Firma Phytec Messtechnik GmbH bietet mit seinem scheckkartengroßen miniMODUL unter Berücksichtigung späterer, speicherintensiver Anwendungen eine Grundlage für ein Prototyping oder eine Kleinserienentwicklung. Neben einem 16bit Mikrocontroller des Typs Infineon C167CR befindet sich eine Anordnung von jeweils 256kByte Flash und SRAM Speicherbausteinen auf einer Grundplatine [Phyt99].

Über zweireihig ausgeführte Stiftleisten auf der Platine wird die Spannungsversorgung und eine Kontaktierung zu allen notwendigen Controller Ein- und Ausgängen sowie den Daten- und Adressleitungen hergestellt. Zusätzlich zu einer Full CAN-Schnittstelle bietet das Modul eine interne und eine externe über einen UART (Universal Asynchronous Receiver/ Transmitter) realisierte serielle Schnittstelle an, s. a. Abbildung 7-11 [Phyt99].

Diese Module kommen aufgesteckt in einer Basisplatine mit integrierter Spannungsversorgung zum Einsatz. Die universelle Basisplatine für das Komfortsystem bietet alle notwendigen zusätzlichen Hardwarebauteile, um die reale Sensorik und Aktorik eines Serienfahrzeugs der PQ24-Plattform bedienen zu können. Dazu gehören im einzelnen vor allem High-Side-Schalter und Brücken zur Ansteuerung der Motoren für die Fensterheber- und Spiegelsteuerung, aber auch die Bustreiber für den CAN-Bus.

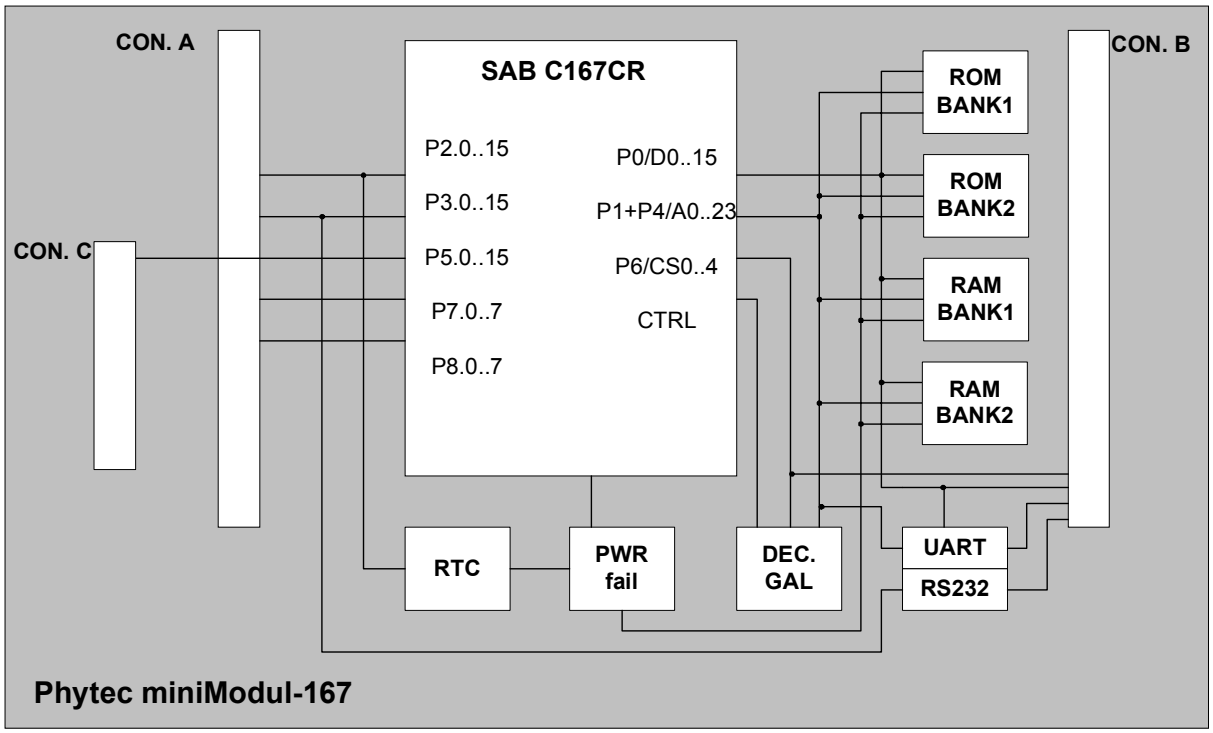


Abbildung 7-11 Schemaschaltbild Phytec MiniModul C167 [Phyt99]

7.4.2.1. Fensterhebermotoransteuerung

Für die Realisierung eines Einklemmschutzes ist die kontinuierliche Messung des Fensterhebermotorstromes notwendig. Dies erfolgt Low-Side, d.h. die Strommessung wird an der massebezogenen Seite der ansteuernden Brückenschaltung durchgeführt. Angelehnt an das Datenblatt des verwendeten elektrischen Schalters BTS 781 GP dient Abbildung 7-12 als Schemaschaltbild.

Für die Diagnose werden neben den normalen Messwerten der Sensoren auch die der Statusleitungen der elektrischen Schalter ausgewertet. Die meisten für den Automotive-Einsatz empfohlenen Treiberelemente, so auch die in den Basisplatinen verbauten, verfügen über zumindest eine Statusleitung, die vom Mikrocontroller ausgelesen werden kann. Wenn ein elektrischer Schalter eine Überlastsituation feststellt, erfolgt durch den Schalter selbst zur Vermeidung von schwerwiegenden Folgefehlern eine Abschaltung der Last und die entsprechende Anzeige anhand der Statusleitung. Die Überlasterkennung ist je nach Schaltermodell als thermische Schutzabschaltung, Überspannungserkennung oder Strombegrenzung ausgeführt, vgl. Abbildung 7-12. Die für die Fensterhebermotordiagnose vorhandenen beiden Statusausgaben des Treiberelementes erlauben eine recht detaillierte Diagnose der anliegenden Motorströme. Es kann hier bereits für jede Motorzuleitung zwischen Normalbetrieb, offener Leitung und thermischer Überlast unterschieden werden.

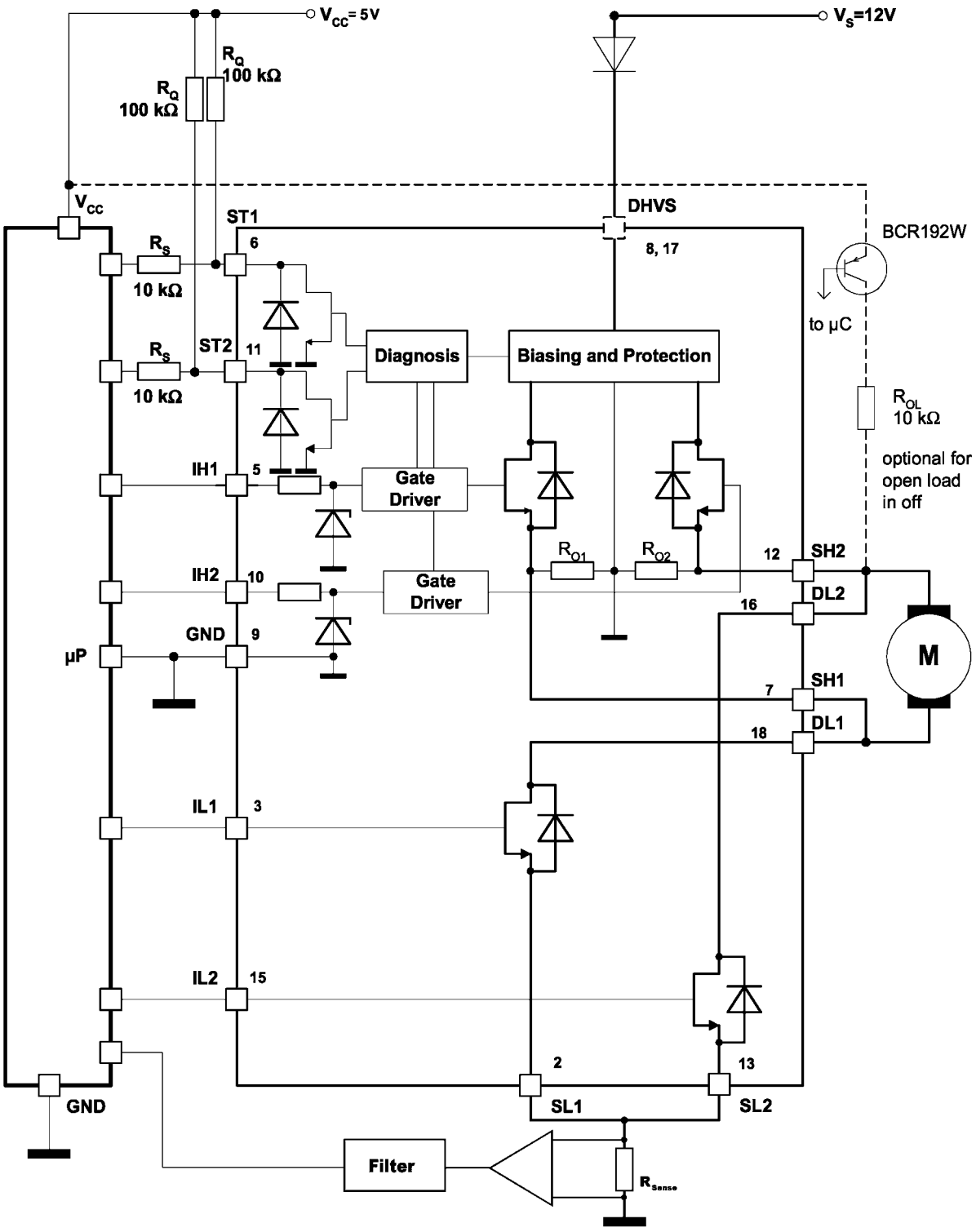


Abbildung 7-12 Ansteuerung des Fensterhebermotors (vgl. [Infi01])

Für den Betrieb des Systems ist die Kenntnis über die gültigen Bereiche des Fensterheberstromes erforderlich. Aus Abbildung 7-13 ist zu ersehen, dass die Ströme im Betrieb je nach Fensterposition und Lastsituation in weiten Bereichen zulässig sind. Ohne Antrieb sollte der Motorstrom wie in den mit A gekennzeichneten Bereichen 0A betragen. Durch die unter-

schiedlichen Widerstände des Fensters im Hoch- gegenüber dem Tieflauf (im Bild mit E bzw. B gekennzeichnet), ergeben sich unterschiedliche Ströme und Fenstergeschwindigkeiten in diesen Fahrsituationen. Beim Motorbetrieb gegen den Endanschlag ergibt sich der höchste Motorstrom (im Bild mit C gekennzeichnet). Fehlereinträge sind erst erlaubt, wenn die Treiberelemente Überlastsituationen erkennen, wie das in den mit D gekennzeichneten Bereichen der Fall ist. Hier wird aufgrund der thermischen Überlast bei einem dauerhaft blockierten Fenster zeitweilig vom Treiberelement die Last abgeschaltet.

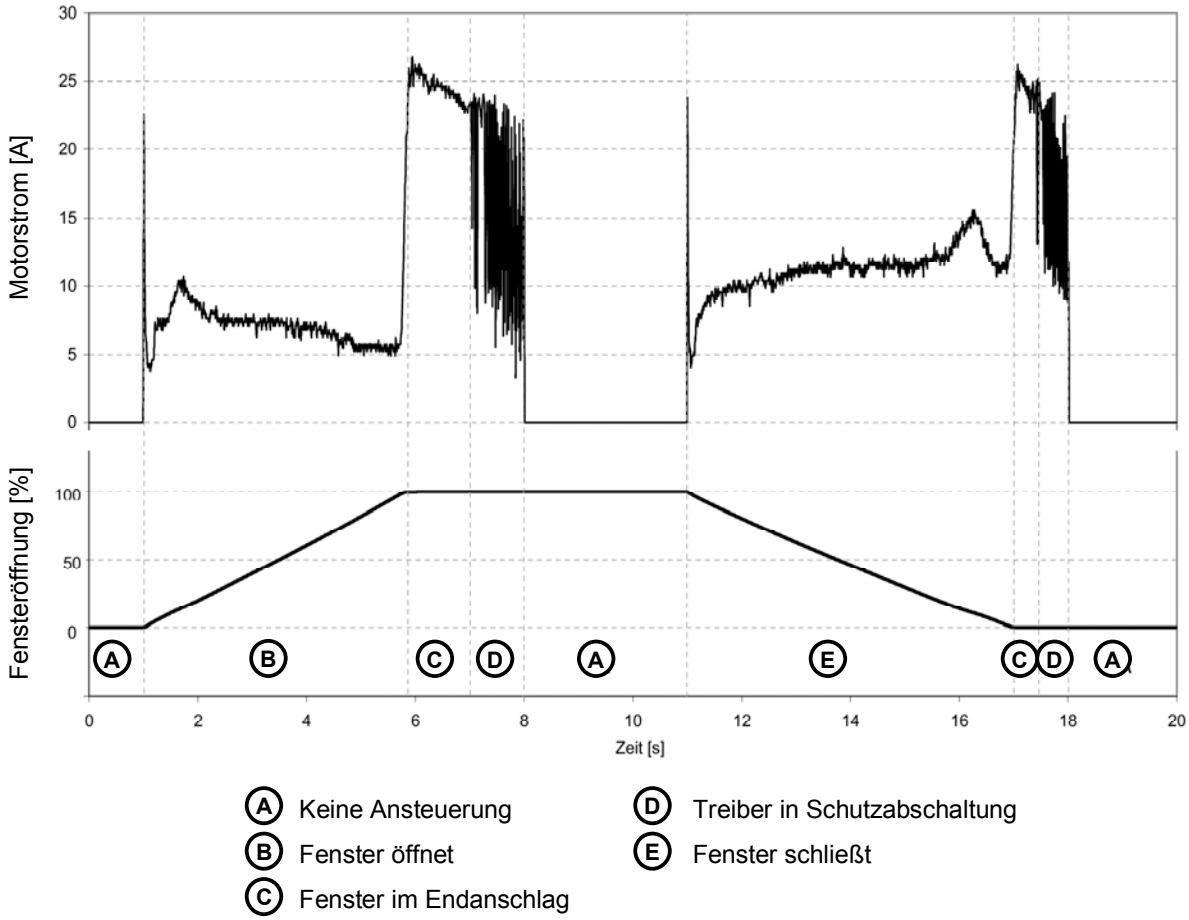


Abbildung 7-13 Fensterlauf und Schutzfunktionen der Treiber

7.4.2.2. Spiegelmotoransteuerung und Heizung

Die zwei Spiegelmotoren für die Einstellung der Spiegelposition in der x/y Ebene werden über lediglich drei Leitungen angesteuert. D.h. es wird für eine Motorzuleitung ein gemeinsames Potential für beide Motoren genutzt. Die Ansteuerung erfolgt über zwei Vollbrückenschaltungen L9997ND wie in Abbildung 7-14 schematisch dargestellt. Da eine Vollbrücke

nur als Halbbrücke zur Spiegelansteuerung betrieben wird, wird die verbleibende Halbbrücke zur Ansteuerung der Spiegelheizung verwendet.

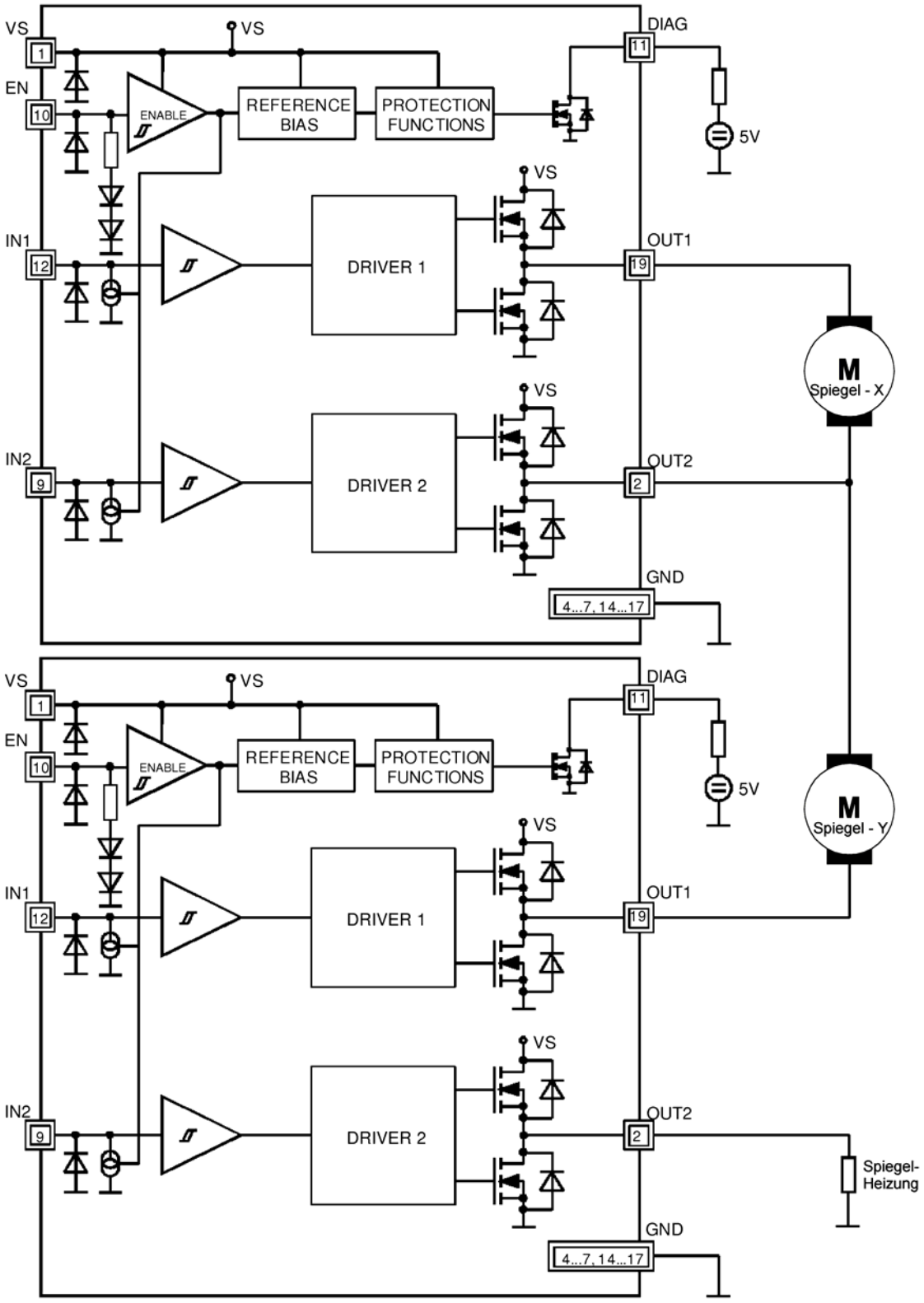


Abbildung 7-14 Ansteuerung von Spiegelflächenmotoren und Spiegelheizung (vgl. [STMi99])

7.4.2.3. Spiegelabklappmotor

Die Spiegelabklappsteuerung ist schaltungstechnisch genau wie die Fensterheberansteuerung gelöst. Unterschied zur Fensterheberansteuerung ist hier lediglich das Weglassen der Strommessung für den Abklappmotorstrom, und mit dem VN770 die Verwendung eines etwas schwächeren Brückenschaltelementes.

7.4.2.4. Bedienfelder

Die Taster und Schalter der Bedienfelder des Komfortsystems im Volkswagen Polo stellen ihren jeweiligen Tast- bzw. Schaltzustand anhand von definierten Widerstandswerten dar. Zur Auslesung durch den Mikrokontroller werden diese mit einem in Reihe geschalteten Referenzwiderstand zur Versorgungsspannung als Spannungsteiler betrieben, wie in Abbildung 7-15 dargestellt. Über eine tabellarische Zuordnung von Widerstandswerten zu Tastzuständen kann das Steuergerät auf die Benutzereingaben reagieren.

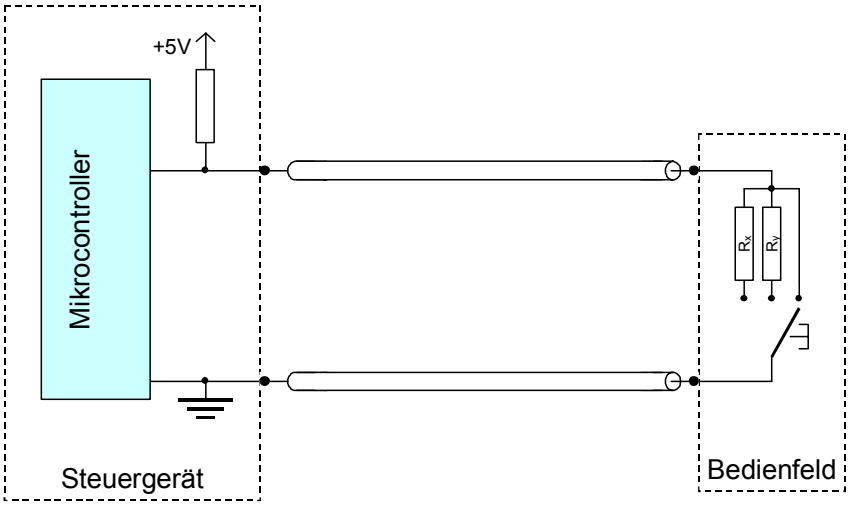


Abbildung 7-15 Prinzipschaltbild Auswertung Tastersignale

Durch diese einfache Anordnung sind alle Diagnosefunktionen bezüglich der Taster- und Schalteranordnungen auf dem Steuergerät selbst durchzuführen. Prinzipiell lassen sich hier lediglich gültige Werte für den gemessenen Widerstand von nicht vorhergesehenen Werten unterscheiden.

7.5. Vergleich der Effizienz mit anderen Diagnosesystemen

Eine im Rahmen der Arbeiten durchgeführte Literaturrecherche hat ergeben, dass die modellbasierte Diagnose in der Zukunft vielfältige Möglichkeiten eröffnen kann. Die Vorteile liegen hier vor allem darin, dass die Entwickler der Diagnose die Abhängigkeiten eines Systems

nicht per Expertenwissen in Diagnosen umsetzen müssen. Bei der modellbasierten Diagnose beschreibt ein Modell das Fahrzeugsystem vollständig und daraus lassen sich die Diagnosen automatisch und damit ebenso vollständig ableiten. Während der Recherche wurden mit OCC'M und ROSE Informatik die beiden einzigen verfügbaren Firmen evaluiert, die kommerziell einsetzbare Software für modellbasierte Diagnose im Automobil anbieten [OCCM03][ROSE03a]. Die Diagnosemodelle werden bei diesen beiden Firmen auf unterschiedliche Weise berechnet. Während ROSE mit seinem Werkzeug RODON mit quantitativen Modellen auf Basis einer Intervallarithmetik arbeitet, setzt OCC'M mit RAZ'R qualitative Diagnosemodelle ein, deren Modellgrößen vom Diagnoseautor in praktikable Zustandsbereiche aufgeteilt werden müssen.

7.5.1. Effizienz bei der Erstellung

Bei allen evaluierten Diagnoseansätzen erfordert das Diagnoseablaufsystem die Erstellung eines eigens erstellten Diagnosemodells des Systems. Darin müssen zusätzlich zur Nennfunktionalität weitergehende Informationen über das korrekte Verhalten des Systems enthalten sein. Bei ROSE und OCC'M wurde dies mit proprietären Entwicklungswerkzeugen ohne Anbindung an bei der Serienentwicklung eingesetzte Entwicklungssoftware realisiert. Die Erstellung der Diagnosemodelle musste daher manuell durchgeführt werden, und erforderte in beiden Fällen exakte Kenntnis der Innen- und Außenbeschaltung der Steuergeräte. Zusätzlich mussten in zeitintensiven Gesprächen zwischen Applikationsentwicklern und Diagnosesystementwicklern die Funktion der Nennsoftware und der eingesetzten Schaltungstechnik erläutert und in dem Diagnosemodell nachgebildet werden.

Unabhängig von der Entwicklung eines quantitativen Diagnosemodells reduzierte auch ROSE Informatik für den Einsatz auf dem eingebetteten System das System auf ein qualitatives Modell, um mit den zur Verfügung stehenden geringen Rechenkapazitäten des Mikrocontrollers auszukommen. Die Abbildung des quantitativen Systems auf ein qualitatives System erfolgte zeitaufwändig im Bereich von Stunden selbst für das im Projekt betrachtete relativ einfache Anwendungsbeispiel aus dem Automobilbereich. Damit ist die Möglichkeit erschwert, der Variantenvielfalt heutiger Fahrzeugentwicklungen adäquat zu begegnen.

Bei Folgeprojekten können bei RODON Modellkomponenten aus einer Bibliothek in die Diagnosemodelle einbezogen werden, was zu einer Verkürzung der Modellierungszeit führt. Dennoch bleibt der hohe Zeit- und Rechenaufwand für die Übersetzung in qualitative Modelle in jedem Fall bestehen.

Für die Anwendung auf dem Steuergerät musste auch OCC'M einen Bearbeitungsschritt durchführen, der eine Optimierung bei der Berechnung der Kandidatenlisten auf dem Steuergerät bewirkt. Dabei werden die Diagnoseregeln auf den optimalen Entscheidungsbaum reduziert.

Es wurde damit in der ersten Phase der Untersuchungen im Rahmen von STEP-X gezeigt, dass automatisiert generierte On-Board-Diagnose anhand von Modellen zwar prinzipiell möglich, aber mit den derzeit verfügbaren Werkzeugen auf den Steuergeräten nicht wirtschaftlich in die Realität umzusetzen ist.

Im Vergleich zu diesen zusätzlich anfallenden Entwicklungsschritten wird beim Vorgehen anhand der in dieser Arbeit vorgestellten Methode ein größerer Anteil von Daten aus dem Entwicklungsprozess mitgenutzt. Besonders relevant ist hier, dass keine zusätzlichen Funktionsmodelle zu erstellen sind. Die bereits in der normalen Funktionsentwicklung implementierten Modelle können zur Ableitung der Abhängigkeitsstrukturen genutzt werden.

7.5.2. Effizienz im Betrieb

Alle exemplarisch angewendeten Diagnosesysteme konnten im Betrieb darstellen, dass die eingestreuten Fehler identifiziert werden. Damit konnte gezeigt werden, dass prinzipiell sowohl modellbasierte als auch strukturbasierte Diagnose in der Praxis angewendet werden kann.

Hinsichtlich der auf dem Steuergerät verwendeten Ressourcen von Speicherbedarf und Prozessorleistung kann keine Aussage getroffen werden.

Ein Vergleich mit konventionellen Werkstattdiagnosesystemen war aufgrund des prototypischen Aufbaus des Demonstrationsobjektes nicht möglich und wurde daher nicht durchgeführt.

8. Zusammenfassung und Ausblick

In dieser Arbeit wird ein Verfahren vorgeschlagen, das die Verbesserung der Qualität und der Effizienz von Diagnosesystemen für Kraftfahrzeuge anstrebt. Die Anwendung des Verfahrens kann mit vergleichsweise geringem Entwicklungsaufwand gegenüber anderen Systemdiagnoseverfahren durchgeführt werden, weil dabei großenteils Daten aus der Funktionsentwicklung mit genutzt werden.

Eine im Rahmen der Arbeiten durchgeführte Literaturrecherche hat ergeben, dass die modellbasierte Diagnose ein großes Potenzial für zukünftige Entwicklungen bietet. Die Entwickler der Diagnose müssen nicht mehr alle Abhängigkeiten eines Systems per Expertenwissen in Diagnosen umsetzen. Anstelle dessen beschreibt ein Modell das System vollständig, aus dem sich Diagnosen automatisch und damit ebenso vollständig ableiten lassen. In dieser Phase wurden mit OCC'M und ROSE die beiden einzigen verfügbaren Firmen evaluiert, die Software für modellbasierte Diagnose anbieten. Auf unterschiedliche Weise werden bei den beiden Firmen die Diagnosemodelle berechnet. Während ROSE mit quantitativen Modellen arbeitet, setzt OCC'M qualitative Diagnosemodelle ein. Als diese Modelle auf einem eingebetteten System berechnet werden sollten, zeigten sich, dass der qualitative Ansatz die einzige Möglichkeit darstellt, überhaupt mit den eingeschränkten Ressourcen eines Steuergerätes auszukommen.

Bei dieser Erprobung wurden zwei wesentliche Schwierigkeiten bei der Anwendung dieser Verfahren offenkundig:

1. die Erstellung des Systemmodells ist aufwendig, und nur unter sehr genauer Kenntnis der eingesetzten Hardware zu realisieren und
2. die Diagnoseregeln konnten nur unter großem Rechenaufwand und unter detaillierter Systemkenntnis erstellt werden. Für die Nutzung in Projekten für den Automobilbereich ist dieser Aufwand derzeit nicht tragbar.

Es wurde damit gezeigt, dass automatisiert generierte On-Board-Diagnose anhand von Modellen zwar prinzipiell möglich, aber mit den derzeit verfügbaren Werkzeugen auf den Steuergeräten nicht in die Realität umzusetzen ist.

In einem alternativen Ansatz wurde die Verbesserung der Diagnosequalität auf einfachere Weise erreicht. Wesentliches Merkmal bei der Definition eines geeigneten praxisnahen Diagnoseverfahrens war die Konzentration auf die Anbindung an den Softwareentwicklungsprozess. Die Lösungen für diese Herausforderungen stellen den Inhalt dieser Arbeit dar.

In der Folge wurden daher unterschiedliche Diagnoseverfahren miteinander verglichen unter dem Aspekt der effizienten Einbindung in den Entwicklungsprozess bei insgesamt guten Diagnoseergebnissen und Beherrschung der Variantenvielfalt. Dabei wurde ermittelt, dass kein bisher bekanntes Verfahren direkt diese Anforderungen erfüllt, und ein neuer Ansatz gefunden werden musste. Diese Aufgabe wurde gelöst, indem aus der Kombination der erkannten Stärken bekannter Diagnoseverfahren ein passendes neues Konzept entwickelt wurde.

Als Einstieg für die neue Diagnoselösung wurde die bei probabilistischen Netzwerken verwendete Systemstruktur als wesentliches Element verwendet. Um eine enge Integration in den Entwicklungsprozess zu erreichen, werden die zugehörigen Strukturelemente anhand von Abhängigkeiten aus den Entwicklungsmodellen automatisiert abgeleitet. Im Gegensatz zu den probabilistischen Netzwerken werden bei dem hier vorgestellten Verfahren keine Ausfallwahrscheinlichkeiten von Strukturelementen betrachtet. Damit bietet sich der Vorteil, dass die Diagnosemodellerstellung beschleunigt wird, weil auf die Sammlung empirischer Daten verzichtet werden kann. Gleichzeitig wird damit die Variantenvielfalt der am Markt befindlichen Fahrzeuge beherrschbar, da individuell für jedes Fahrzeug ein passendes Systemmodell erstellt werden kann, ohne auf Vollausstattungsmodelle oder ähnliches als Referenzmodelle zurück greifen zu müssen. Eine weitere Änderung gegenüber klassischen probabilistischen Netzwerken besteht in der Einführung der Rückkopplung zwischen den Strukturelementen aufgrund der Tatsache, dass in den heutigen vernetzten Systeme umfangreiche komplexe Wechselwirkungen zwischen Funktionen realisiert sind.

Das vorgestellte Systemdiagnoseverfahren ist in der Realisierung in der Werkstatt flexibel. Eine Erstellung der notwendigen Daten in der Werkstatt kann auch auf Basis von Strukturinformationen geschehen, die auf den Steuergeräten abgelegt ist. In dieser Form wird ein immer aktuelles Bild der Systemstruktur für jede Werkstatt zugänglich

Mit dem geschilderten Vorgehen kann auch der Test des Systems effizienter gestaltet werden, da durch die Kenntnis von Strukturdaten der Testaufwand vermindert werden kann. In diesem Zusammenhang entstand die Definition der Beschreibung von Hardwareschnittstellen des entwickelten Systems . Aufgrund der automatisiert lesbaren Hardwareschnittstellenbeschreibungen können diese Daten gleichermaßen von Test und Diagnose genutzt werden.

Die Verwendung der Strukturinformationen zur Diagnose eines automobilen Systems führte in den betrachteten Fällen zu einer Verbesserung bzw. einer Einstellung der bisherigen Diagnoseergebnisse. Dennoch sollte beachtet werden, dass das bisherige Wissen des Werkstattper-

sonals in dem vorliegenden Verfahren nicht einfließt. Dieser Punkt kann und sollte in folgenden Arbeiten berücksichtigt werden.

In Folgearbeiten ist auch die praktische Bedeutung der Untersuchungen weiter zu belegen, da hier aufgrund der universitären Ansiedlung des Projektes STEP-X nur auf Basis von prototypischen Systemen die qualitative Realisierbarkeit demonstriert werden konnte. Bei den für den Serieneinsatz notwendigen Erweiterungen des Diagnosesystems ist die Integration von Ausfallwahrscheinlichkeiten anhand von Felddaten vorstellbar, um die Zielführung des Diagnosesystems weiter zu optimieren.

Im Rahmen der Bearbeitung wurden bisher in der Literatur getrennt voneinander betrachtete Aspekte von Diagnoselösungen kombiniert und in einen logischen Gesamtzusammenhang gebracht. Dabei wurden einfach abgeleitete kausale Abhängigkeiten als fahrzeugweit integrierbares Element für ein Diagnosesystem identifiziert und genutzt. Im Gegensatz zu bisherigen Betrachtungen können hierbei in einem einzigen logischen Kontext physikalisch völlig unterschiedliche Zusammenhänge identisch ausgewertet werden. Damit wird das Diagnosesystem in die Lage versetzt, Kommunikationsabhängigkeiten, Softwareabhängigkeiten und elektrische Abhängigkeiten kombiniert zu betrachten.

Diese Kombination aus umfassender Betrachtung von Strukturen aus Hardware, Software und Kommunikation bei gleichzeitiger enger Integration in den Entwicklungsprozess ist ein Ergebnis dieser Arbeit und wurde bisher in dieser Form nicht vorgestellt.

9. Literatur

[ASAM06] ASAM e.V.; http://www.asam.de; 10.3.2006

[Auto04] Automobil Produktion; Schafft Volkswagen den Spagat?, 12.2004

[Auto07] AUTOSAR; http://www.autosar.org; 16.4.2007

[AVLD05] www.avlditest.com; Produktbroschüre AVL DiScan 8000 – Faultcode reader with information system and oscilloscope; 21.11.2005

[Ayeb00] Ayeb, M.; Lichtenthäler, D.; Theuerkauf, H.; Neuronale Netze für die On-Board-Diagnose, 2000

[Bäke00] Bäker, B.; Heinzelmann, A.; Luka, J.; Rehfus, B.; Systemdiagnoseverfahren und Vorrichtung zur Durchführung eines Systemdiagnoseverfahrens; Offenlegungsschrift DE10051781A1, 2000

[Beil96] Beil, F.; Schürmann, B.; Integration der Bordnetzfunktionen – Ein Schritt zu einem Gesamtkonzept der Fahrzeugfunktionen, VDI-Tagung Elektronik im Kraftfahrzeug, 1996

[Benn82] Bennet, J. S.; Hollander, C. R.; DART: An Expert System for Computer Fault Diagnosis; Proceedings of the 5th International Joint Conference on Artificial Intelligence, Vol. 2, 1982

[Bikk02] Bikker, G.; Schroeder, M.; Methodische Anforderungsanalyse und automatisierter Entwurf sicherheitsrelevanter Eisenbahnleitsysteme mit kooperierenden Werkzeugen; Dissertation, Braunschweig, 2002

[Bogd91] Bogdanoff, M.; Hada, S.; Overview of On-Board Diagnostic Systems Used on 1991 California Vehicles; Warrendale, Pennsylvania, 1991

[Borg04] Borgelt, C.; Kruse, R.; Probabilistische grafische Modelle und ihre Anwendung in der Automobilindustrie, Datenbank-Spektrum, 2004

[Bosc03] Bosch; Computer Aided Service CAS plus von Bosch – Noch effektivere Fehlersuche mit Werkstatt-Software Esitronic; Presseinformation; 2003

[Bosc04] Bosch; Einfache Lösung für die komplexe Fahrzeugdiagnose - Die neue „Fahrzeug-System-Analyse FSA" von Bosch; Presseinformation; 2004

[Brüg00] Brügge, B.: Object-Oriented Software Engineering. Prentice-Hall, 2000

[Cosf95] Cosfeld, R.; Hohenner, H.; Niggemeyer, H.; Möglichkeiten und Grenzen elektronischer Motorsteuerungen; 20. Fortschrittsbericht, VDI Verlag, Reihe 12, Nr. 239, Bd. 2; 1995

[Curt91] Curth, M.; Böscher, A., Raschke, B.; Entwicklung von Expertensystemen; Hanser Verlag, München, 1991

[Daim05] Daimler Benz; EPC / WIS / ASRA Werkstattinformationssystem; Produktbeschreibung, 2005

[DeKl86] DeKleer, J.; An assumption based truth maintenance system; Artificial Intelligence 28, 1986

[DeKl87] DeKleer, J.; Williams, B.; Diagnosing multiple faults; Artificial Intelligence 32, 1987

[DIN25424] DIN 25424, Teil 1, Fehlerbaumanalyse - Methode und Bildzeichen, Beuth Verlag, Berlin, 1981

[DIN55350] DIN 55350 – Begriffe zu Qualitätsmanagement und Statistik, 1985

[Drös00] Dröschel, W.; Wiemers, M.; Das V-Modell 97; Oldenbourg, 2000

[Dude04] Dudenhöffer, F.; Krüger, M.; Schmaler, H.; Qualitäts-Herausforderung - Stabiles Energiemanagement; ATZ, 2004

[Felb05] Felbinger, L.; Schaal, H.-W.; Trucks unter Kontrolle – CAN und offene Protokolle im Nutzfahrzeug, elektronik industrie, 10.2005

[Fija02] Fijany, A;. Vatan, F.; Barrett, A.; Maxkey, R.; New Approaches for Solving the Diagnosis Problem; IPN-Progress Report 42-149, 2002

[FORS05] http://www.forsoft.de/, 27.4.2005

[Frei93] Freitag, H.; Methoden zur modellbasierten Diagnose von Systemen mit komplexer Struktur und zeitabhängigem Verhalten; Dissertation Universität Hamburg, 1993

[Frit96] Fritz, R.; Runge, W.; Anforderungsprofile an Entwicklungsingenieure - Anpassung an neue Formen der Zusammenarbeit sowie an geänderte Arbeitsprozesse und –inhalte; Tagung Elektronik im Kraftfahrzeug, VDI Verlag, Bericht 1287,1996

[Görz00] Görz, G.; Rollinger, C.-R.; Schneeberger, J.; Handbuch der künstlichen Intelligenz, Oldenbourg, München 2000

[Groc93] Grochtmann, M.; Grimm, K.; Classification Trees for Partition Testing; Software Testing, Verification & Reliability, vol. 3, no. 2, 1993

[Habe05] Habel, C.; Eschenbach, C.; Wissensrepräsentation; Vorlesung; 2005

[Harm03] Harms, M.; Horstmann, M.; Interlacing of Diagnosis and Test with Development of Embedded Systems; Seminar „Software Intensive Embedded Systems – with Special Emphasis on Automotive“; Vortrag, 2003

[Hart01] Hartmann, N.; Automation des Tests eingebetteter Systeme am Beispiel der Kraftfahrzeugelektronik; Dissertation; 2001

[Hebb49] Hebb, D. O; The organization of behavior: A neuropsychological theory; Wiley, 1949

[Hein99] Heinzelmann, A.: Produktintegrierte Diagnose komplexer mobiler Systeme; Dissertation, Universität GH Paderborn, 1999

[Hors05] Horstmann, M.: Verflechtung von Test und Entwurf für eine verlässliche Entwicklung eingebetteter Systeme im Automobilbereich, Dissertation Technische Universität Braunschweig, 2005

[Infi01] Infineon Technologies; TrilithicIC BTS 781 GP, Target Data Sheet; 2001

[ISO14230] ISO 14230; Road Vehicles — Diagnostic Systems – Keyword Protocol 2000; 1996

[ISO15765] ISO 15765; Road Vehicles — Diagnostic Systems – Diagnostics on CAN, 2001

[ISO8402] DIN EN ISO 8402, Quality management and quality assurance – Vocabulary, 1995

[ISO9141] ISO 9141; Road Vehicles – Diagnostic Systems – Requirements for Interchange of Digital Information; 1989

[Jers03] Jersak M.; Richter, K.; Racu. R;, Staschulat, J; Ernst, R.; Braam, J.; Wolf, F.; Jerraya, A.; Yoo, S.; Verkest, D.; Formal methods for integration of automotive software in Embedded Software for SoC, Kluwer Academic Publishers, 2003

[John03] John, B; Softwareentwickler arbeiten an der Vernetzung aller Werkstattsysteme; Automobilwoche, 2003

[Kahn88] Kahn, G., S.; MORE: From Observing Knowledge Engineers to Automating Knowledge Acquisition; Automating Knowledge Acquisition for Expert Systems; Boston, Kluwer Academic Publishers, 1988

[Lang05] Lange, O. – http://ottolange.twoday.net, 17.5.2005

[Lehm00] Lehmann, E.; Wegener, J.: Test Case Design by Means of the CTE XL; Proceedings of the 8th European International Conference on Software Testing, Analysis & Review (EuroSTAR 2000), Kopenhagen, 2000

[Lemm94] Lemmer, K.; Diagnose diskret modellierter Systeme mit Petrinetzen; Braunschweig, Dissertation 1994

[Luek04] Lüke, S.; Dezentraler Diagnoseansatz für dynamische und mechatronische Systeme; Dissertation Technische Universität Dresden; Shaker Verlag,

[Mach93] Macho, S.; Modelle des Lernens: Neuronale Netze; Vorlesung Universität Freiburg, 1993

[Marb91] Marburger, H.; Wissenserwerb für technische Diagnosesysteme; Qualität und Zuverlässigkeit, Band 36 Heft 9, 1991

[Maso05] Mason, M.; O'Neill, K.; FPGA Reliability in Space-Flight and Automotive Applications; http://www.fpgajournal.com/articles_2005/20050906_actel.htm; 6.9.2005

[Mega05] Megacomm!; http://www.megacomm.de/produkte/megadiag.htm; Wolfsburg, 16.11.2005

[Merc04] Future Automotive Industry Structure (FAST) 2015; Studie; Mercer Consulting und Fraunhofer Gesellschaft, 2004

[Miss96] Misselwitz, S. D.; Seibold, W.; Simulation and Diagnosis by means of a single model; Proc. Of the 29th ISATA Conference, Italien, 1996

[Müll04] Müller, T.; Entwurf und Implementierung eines steuergeräteübergreifenden Diagnosesystems auf Basis von Hardwarestrukturinformationen; Diplomarbeit, Braunschweig, 2004

[Mutz03] Mutz, M.; Harms, M.; Horstmann; et. al.; Ein durchgehender modellbasierter Entwicklungsprozess für elektronische Systeme im Automobil; VDI Kongress Elektronik im Kraftfahrzeug, Baden-Baden, 2003

[Mutz05] Mutz, M.; Eine durchgängige modellbasierte Entwurfsmethodik für eingebettete Systeme im Automobilbereich; Technische Universität Braunschweig, Cuvillier Verlag, 2005

[OCCM03] http://www.occm.de/, 20.3.2003

[Oliv03] Olive, X.; Trave-Massuyes, L.; Poulard, H.; Variant methods for automatic generation of near optimal diagnosis trees; Fourteenth International Workshop on Principles of Diagnosis, 2003

[OSEK01] OSEK/VDX Binding Specification Version 1.3; 2001

[Pete05] Peter, C.; Penzenstadler, F.; ASAM Automotive Electronics Primer; Vortrag; GM-TechDay, 2005

[Phyt99] PHYTEC Technologie Holding AG; miniMODUL 167 – Hardware Manual; Handbuch, 1999

[Pisc04] Pischetsrieder, B.; Vorstellung der Volkswagen AG Bilanz 2003; Wolfsburg, 2004

[Pric63] Price, R.; An Essay towards solving a Problem in the Doctrine of Chances, Brief, 1763

[Rich93] Richter, M.; Pfeifer, T.; .Diagnose von technischen Systemen - Grundlagen, Methoden und Perspektiven der Fehlerdiagnose; Deutscher Universitätsverlag, 1993

[Rett05] Retter, J.; Schmidts, U.; Advantages of the Frontloading Process in Developing ECU Diagnostics; EPN automotive electronics; 2005

[Ring99] Ringel, T.: Analyse, Akquisition und Verarbeitung distribuiert vorliegenden Wissens; Dissertation Technische Universität Hamburg-Harburg, 1999

[ROSE03] R.O.S.E. Informatik, RODON 3 Benutzerhandbuch, 2003

[ROSE03a] http://www.rose.de/, 10.3.2003

[Rull01] Rullhusen, I.; Probabilistische Wissensrepräsentation mit Bayes-Netzen, 2001

[Rupp01] Rupp, C.; Requirements-Engineering und –Management; Hanser Verlag, 2001

[Russ03] Russel, S.; Norvig, P.; Artificial Intelligence – A Modern Approach; Prentice Hall, 2003

[Raun02] Rauner F.; Hitz, H.; Spöttl, G.; Becker, M.; Aufgabenanalyse für die Neuordnung der Berufe im Kfz–Sektor; Abschlussbericht, Bremen und Flensburg, 2002

[Scha05] Scharnhorst, T.; AUTomotive Open System Architecture (AUTOSAR) – An Industry-wide Initiative to Manage the Complexity of Emerging E/E Architectures, Genf, 2005

[Schi97] Schiller, F.; Diagnose dynamischer Systeme auf der Grundlage einer qualitativen Prozeßbeschreibung; Erlangen; 1997

[Schn02] Schnieder, E.; Integration heterogener Modellwelten der Automatisierungstechnik; Workshop „Modelle, Werkzeuge und Infrastrukturen zur Unterstützung von Entwicklungsprozessen“, Aachen, Wiley-VCH Verlag, 2002

[Schr05] Schroeder, P.; Zap! How to Diagnose and Solve Automotive Electrical Problems; Grassroots Motorsports; 2005

[Seem00] Seemann, J.; Wolff, J.; Softwareentwurf mit UML; Springer Verlag, 2000

[Spit01] Spitzer, B.; Modellbasierter Hardware-In-The-Loop Test von eingebetteten elektronischen Systemen; Dissertation; 2001

[Stef04] Steffelbauer, M.; Standardisierter Diagnoseprozess nach ASAM; elektronik automotive 4/2004, 2004

[Stein94] Stein, W.; Objektorientierte Analysemethoden; Wissenschaftsverlag, 1994

[STMi99] ST Microsystems; L9997ND, Datasheet; 1999

[Stru89] Struss, P.; Dressler, O.; Physical Negation: Integrating fault models into the general diagnostic engine; Joint Conference on Artificial Intelligence (IJCAI-89), Detroit, 1989

[Tecn05] Tecno GmbH; http://www.tecnogmbh.de; Produktbeschreibung Tecno Reflex 3130; 10.11.2005

[VDI85] Verein deutscher Ingenieure; VDI 2880 – Speicherprogrammierbare Steuerungsgeräte – Programmier- und Testeinrichtungen; VDI Verlag, Düsseldorf, 1985

[Vect05] Vector Informatik GmbH; Candela – Candela Studio 4.0; Produktbeschreibung; 2005

[VW97] Internes VW-Dokument; Fahrzeugdiagnose-, Meß-, und Informationssystem VAS 5051; Selbststudienprogramm 202; 1997

[Wein05] Weinfurther, J.; Automotive Diagnostic and Security System Issues; Vector Congress, Troy, Michigan; 2005

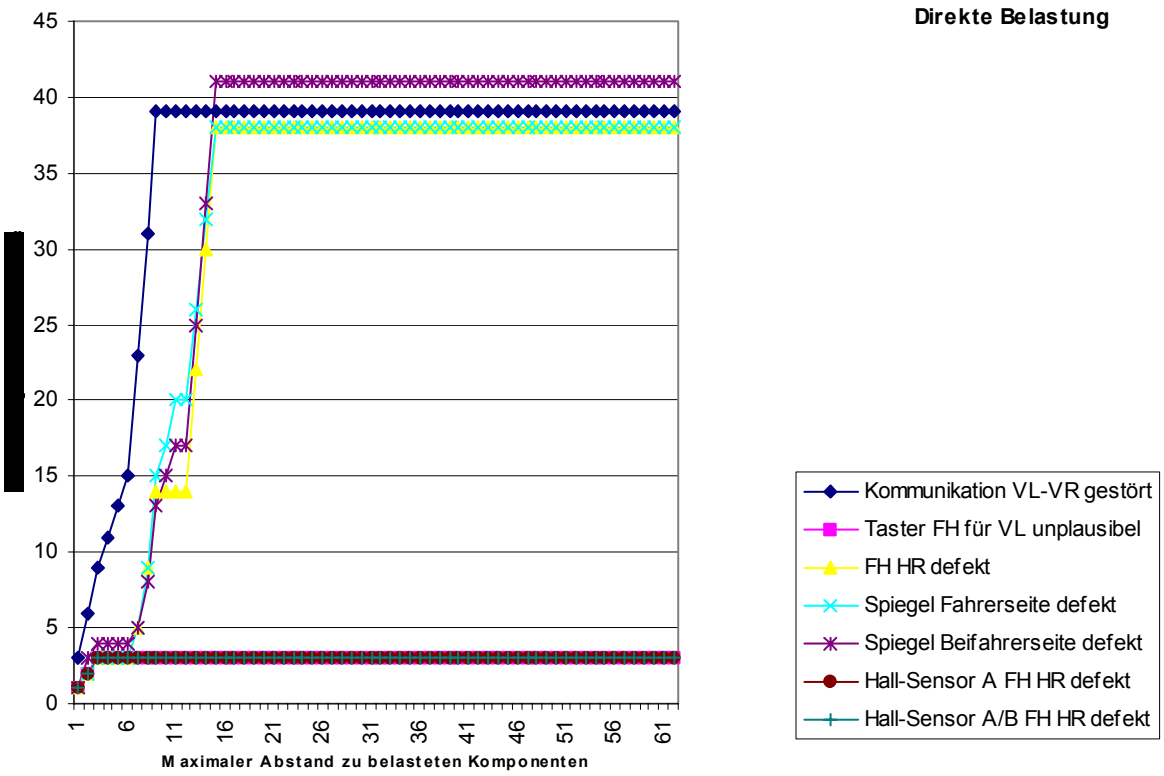
[Wiki05] Wikipedia; Der Begriff Mechatronik; http://de.wikipedia.org/wiki/Mechatronik; 22.11.2005

[Würt05] Würth Online World GmbH; WoW! Diagnose Software; http://www.wow-portal.com; 6.9.2005

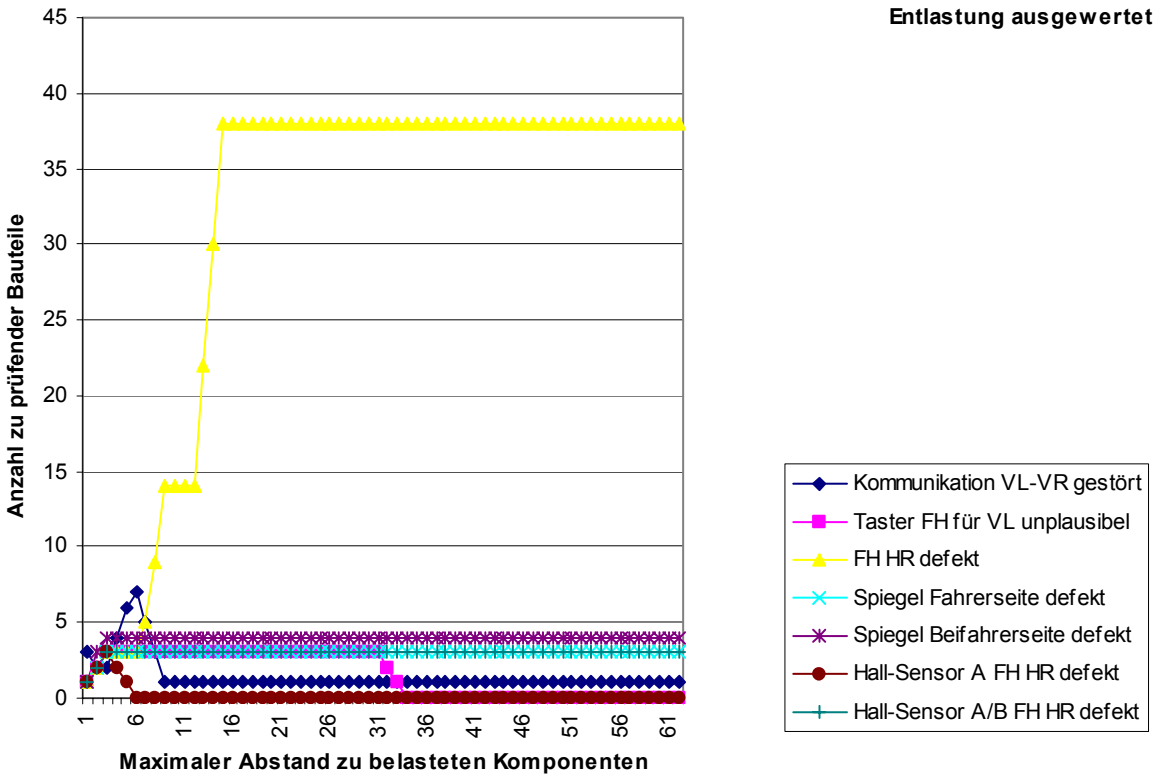
10. Anhang

10.1. Diagnosekandidaten am STEP-X Demonstrationsobjekt

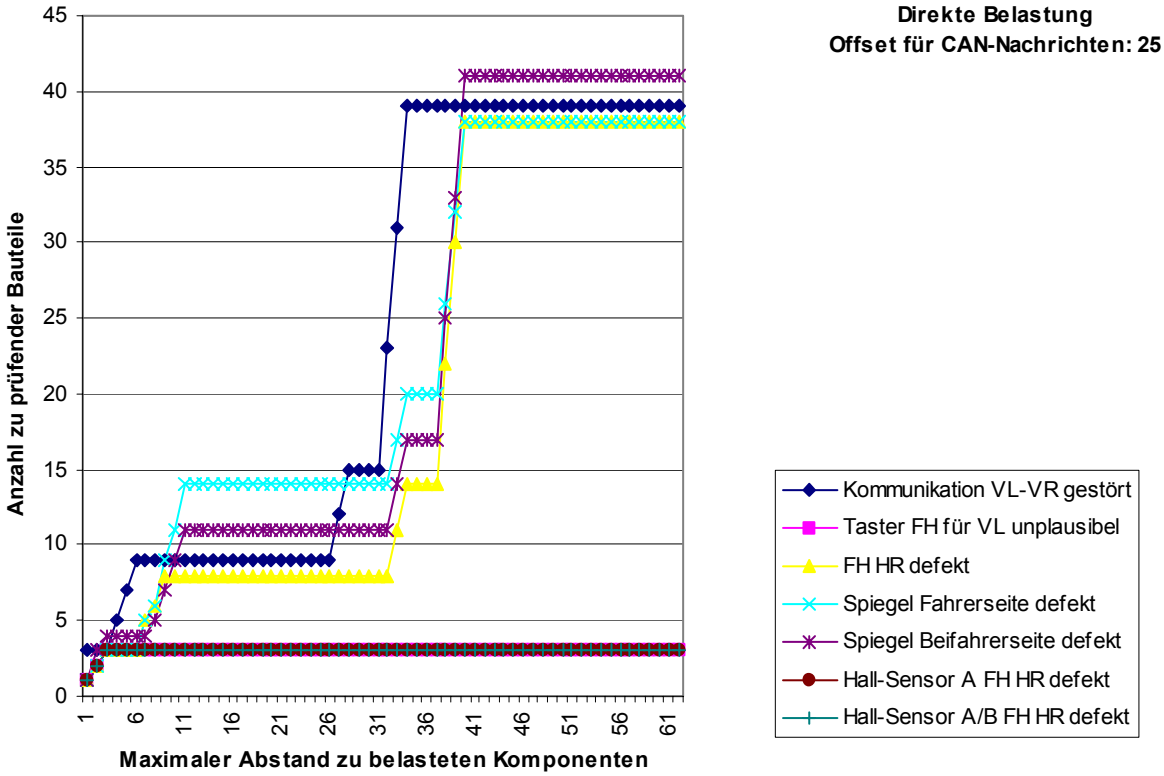
10.1.1. Unbedingte Ausbreitung von Fehlern



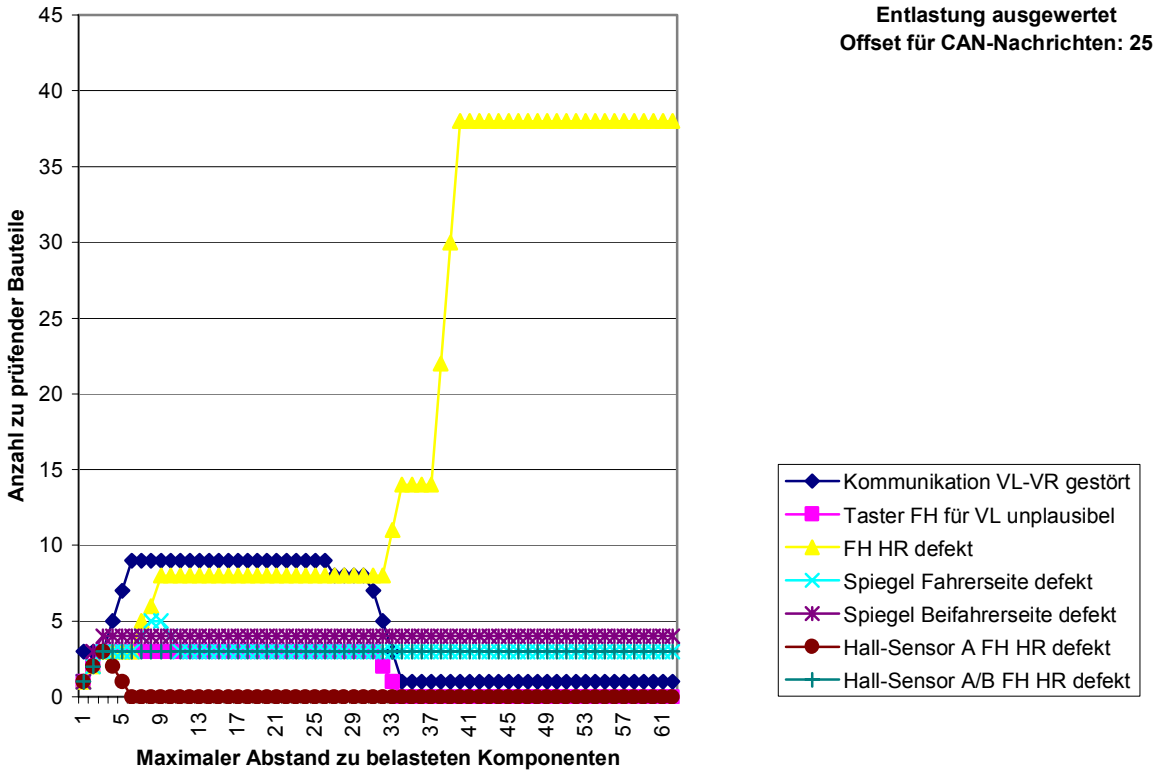
10.1.2. Ausbreitung bei Anwendung der Entlastungsfunktion



10.1.3. Ausbreitung bei Anwendung des CAN-Kosten-Offsets



10.1.4. Anwendung von CAN-Kosten-Offset und Entlastungsfunktion



11. Bedienung des Strukturexport-Programms

11.1. Übersicht über das Programm

Das Strukturexport-Programm liest Strukturen ein und berechnet die Abhängigkeiten der Elemente untereinander. Die verarbeiteten Strukturen können zu Diagnose- und generellen Auswertezwecken weiter verwendet werden.

11.2. Einlesen von Strukturen

Zunächst müssen Strukturdaten in das Programm eingelesen werden. Dies geschieht über den Menüpunkt Strukturdaten->Strukturdaten hinzufügen. Es öffnet sich ein Fenster wie in Abbildung 11-1. Es können über den Butten „Strukturdaten lesen“ in beliebiger Folge Strukturdaten analysiert und zu dem Gesamtmodell hinzugefügt werden. Wenn alle Strukturdaten eingelesen sind, wird das Fenster durch Klick auf den „OK“-Button geschlossen.

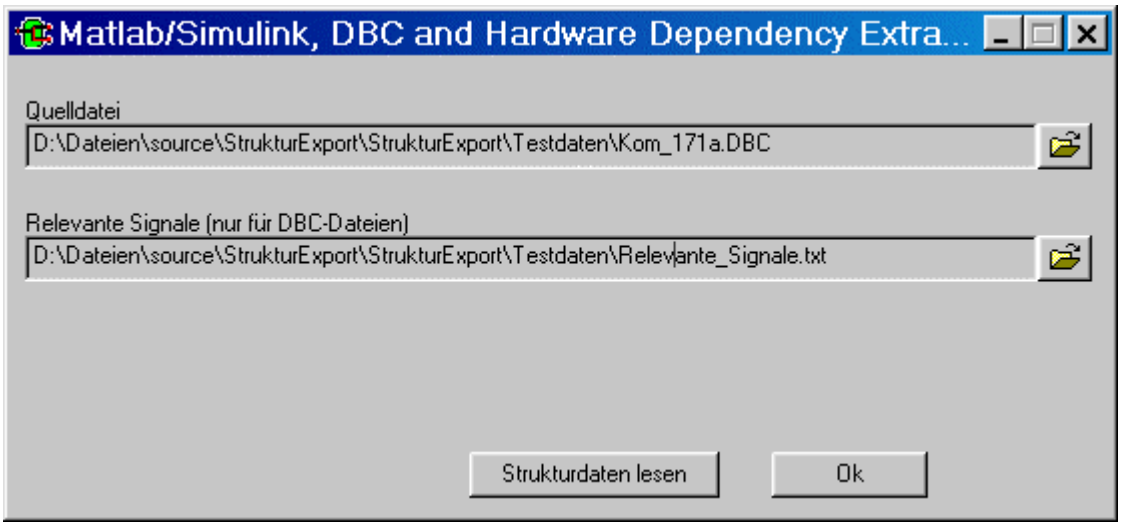


Abbildung 11-1 Dialogfeld "Strukturdaten einlesen"

Gültige Einleseformate sind:

- Simulink-Dateien (*.mdl)
- CAN-Datenbasen (Vector *.dbc)
- DOORS-Hardware-Exporte (*.csv)

11.2.1. Simulink-Dateien

Aus den Simulink-Dateien wird die Software-Struktur ausgewertet. Bei Simulink-Dateien ist zu beachten, dass die Modelle möglichst keine Bibliothekselemente enthalten, da diese nicht

tiefer verfolgt werden können. Durch Aufbrechen der Links zu den Bibliothekselementen in der Simulink-Umgebung können die Bibliothekselemente in das Modell so integriert werden, dass eine Strukturauswertung möglich ist. Es werden lediglich Elemente in die Gesamtstruktur übernommen, die den Bezeichner-Präfix „IN_“ bzw. „OUT_“ haben.

11.2.2. CAN-Datenbasen

Zusätzlich zur Datenbasis muss im Feld Relevante Signale eine Text-Datei angegeben sein, die die für die Strukturauswertung gültigen Signale definiert. Lediglich die Struktur, die aus den relevanten Signalen generiert wird, wird in die Gesamtstruktur übernommen. Auch die Anpassung zwischen den in der CAN-Datenbasis verwendeten Steuergerätenamen und den aus anderen Quellen abgeleiteten Strukturelementen wird in dieser Datei erwartet.

11.2.3. DOORS-Hardware-Export

In der Spezifikation (DOORS) werden die elektrischen Verbindungen des Systems abgelegt. Die DOORS-Daten beschreiben die elektrischen Verbindungen des Systems.

Die Zuordnung zwischen Fehlercodes und belasteten Struktur-Elementen ist Teil der DOORS-Hardware-Export-Dateien. Ebenso wird hier die Verbindung zwischen Hardware und Software definiert. Zusätzlich wird der Bezug zwischen Fehlercode und belastetem Strukturelement hier beschrieben.

In die Gesamtstruktur fließen die Hardwarebeschreibung ohne die Leitungsbezeichnungen und die Verbindungen zwischen Hardware und Software. Auch die Fehlerdefinitionen werden integriert.

11.3. Ausgabedateien des Programms

Das Ergebnis nach dem Einlesen einer Modell-Datei sind zwei Matrizen (<model>_matrix1.txt, <model>_matrix2.txt), die die ausgewertete Struktur der Software enthält. In Matrix 1 sind lediglich die Direktverbindungen zwischen den Elementen eingetragen, während in der Matrix 2 auch die Abhängigkeiten zwischen weiter entfernten Strukturelementen eingetragen ist. Die Verbindungen, die in der Datei Matrix 1 eingetragen sind, werden in ausführlicher Textnotation zusätzlich in der Datei <model>_struct.txt ausgegeben.

Weitere Ausgabe des Einlesens ist die Eingabe-Ausgabe-Abhängigkeitsmatrix (<model>_essence.txt). Diese Datei enthält nur die Beziehungen zwischen den in die Gesamtstruktur übernommenen Elementen.

Über den Menüpunkt „Strukturdaten->Strukturdaten speichern“ kann die Gesamtstruktur gespeichert werden, die aus den Teilmodellen zusammen gesetzt wurde. Die Datei enthält in der Matrixnotation die Abhängigkeiten der Elemente untereinander mit Entfernungsangaben zwischen indirekt verbundenen Elementen.

„Strukturdaten->Graph speichern“ speichert die ermittelten Abhängigkeiten in einer LEDA-Graph-Datei für die mögliche Visualisierung der Abhängigkeitsstruktur.

11.4. Nutzungsmöglichkeiten der Gesamtstruktur

Die Gesamtstruktur kann innerhalb des Programms für einfache Analysen herangezogen werden. Über den Menüpunkt „Strukturdaten->Abhängigkeiten zeigen“ wird in einem Fenster wie in Abbildung 11-2 aufgezeigt, welchen Einfluss die Strukturelemente innerhalb der Struktur haben. Dies kann dafür genutzt werden, die Diagnostizierbarkeit eines Systems zu bewerten.

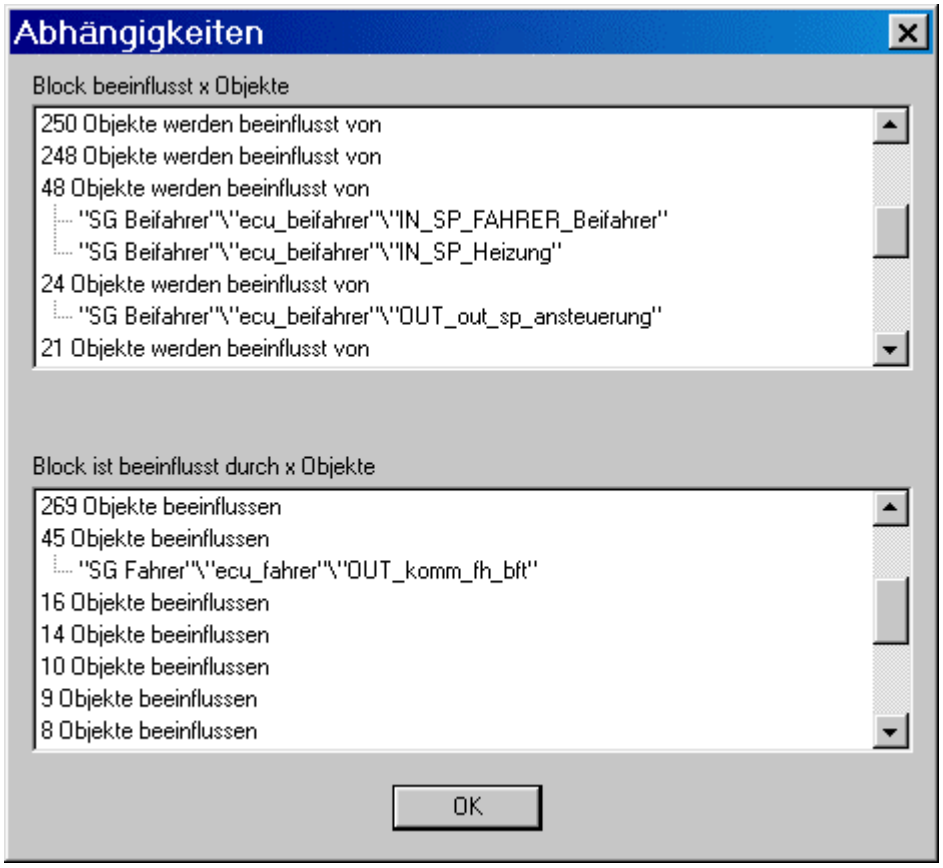


Abbildung 11-2 Abhängigkeiten der Strukturelemente

Im oberen Teil des Fensters wird gezeigt, wie viele Objekte von einem bestimmten Objekt abhängig sind, d.h. auf wie viele Objekte sich ein Fehler dieses einen Objektes auswirken kann.

Im unteren Teil ist dargestellt, wie viele Objekte ein bestimmtes Objekt beeinflussen. Damit wird gezeigt, wo im System sich Objekte befinden, die von sehr vielen unterschiedlichen Fehlern beeinflusst werden können.

Die Menüpunkte Verbindungen und Diagnose funktionieren nur, wenn Fehlercodebeziehungen aus den DOORS-Exporten geladen werden. Anderenfalls lassen sich keine Strukturelemente mit Fehlern belasten.

11.5. Menüpunkt Fahrzeugsystemdiagnose

Unter dem Menüpunkt Fahrzeugsystemdiagnose kann der Fahrzeugstatus anhand von aufgezeichneten Daten verändert werden, und daraufhin die Systemdiagnose durchgeführt werden.

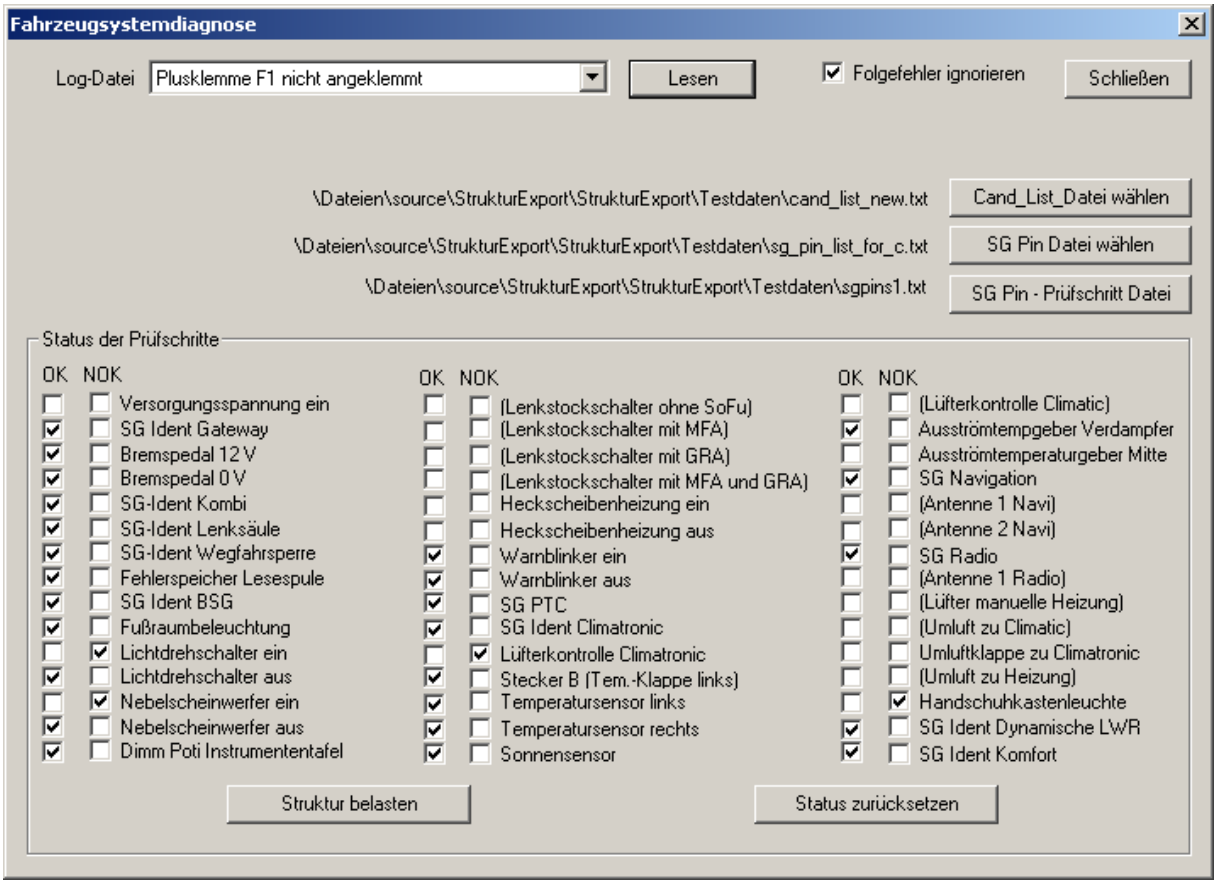


Abbildung 11-3 Fahrzeugsystemdiagnose-Dialog

Zusätzlich zu dem ursprünglichen Umfang wurden die Auswahlmöglichkeiten bei der Diagnose ein wenig verfeinert. Über das Drop-Down-Menü „Log-Datei“ lassen sich die damals

erfolgten Diagnosefälle auswählen. Über die Checkbox „Folgefehler ignorieren“ lässt sich auswählen, ob die Strategie, beim Auftreten von bestimmten Fehlern einzelne Prüfschritte zu ignorieren angewendet wird.

Nach einem Klick auf den Button „Lesen“ erfolgt die Ausgabe der Diagnosen im Strukturexport-Hauptfenster (Abbildung 11-4).

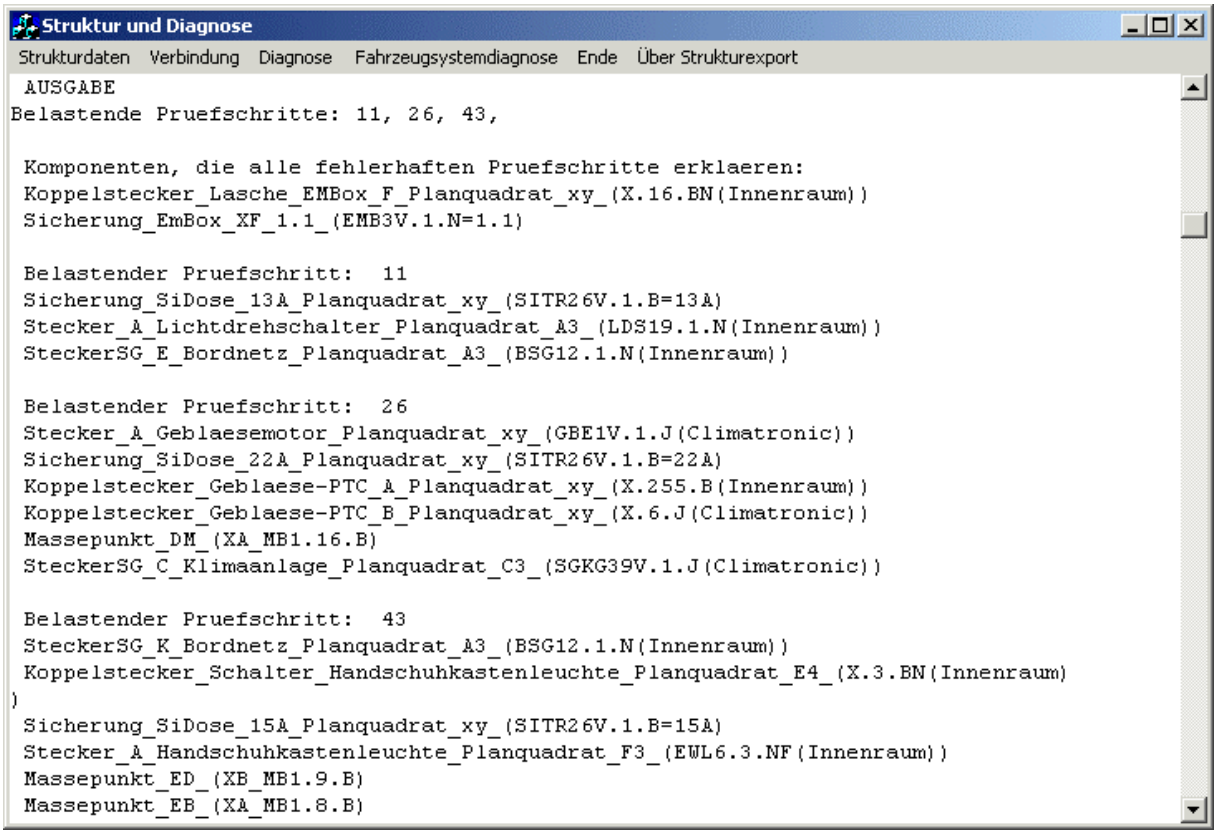


Abbildung 11-4 Diagnoseausgabe

11.6. Beschreibung der Ausgabeformate der Matrizen

Das Ausgabeformat für Matrizen aus dem Strukturexportprogramm ist das CSV-Format. D.h. eine Matrix wird in Form einer Texttabelle gespeichert, wobei die Spalteneinträge durch ein Semikolon getrennt sind, und Tabellenzeilen durch Zeilen innerhalb der Datei definiert sind.

Eine Zeile der Tabelle ist in folgender Form abgespeichert:

```
<ELEMENTTYP>:<ELEMENT>;<DISTANZ1>;<DISTANZ2>;...;<DISTANZn>
```

Zwischen den einzelnen Tabelleneinträgen kann eine beliebige Anzahl von Leerzeichen eingefügt werden, um die Lesbarkeit mit einfachen Texteditoren zu erhöhen.

Beispiel:

```
Inport:\\"ECU_1"\"IN_Taster".Out1      ;0;1;0;1;0;0
```

Zur Erklärung der einzelnen Einträge:

<ELEMENTTYP> bezeichnet den Typ des Elementes. Erlaubt sind beliebige Zeichenfolgen bestehend aus der Zeichenmenge [`0´..`9´, `a´-`z´, `A´-`Z´, `"´, `-´]. Der Elementtyp wird z.B. aus Simulink-Modell-Komponenten übernommen. Im Beispiel lautet er Inport.

<ELEMENT> bezeichnet den Pfad und das Element. Erlaubt sind beliebige Zeichenfolgen bestehend aus der Zeichenmenge [`0´..`9´, `a´-`z´, `A´-`Z´, `"´, `-´]. Hierarchieebenen des Pfades zum Element werden durch das Zeichen `\´ angelehnt an die Schreibweise von Windows-Suchpfaden notiert. Der Elementbezeichner enthält einen Postfix in der Form .Inx bzw. .Outx um die Art des Elementeportes als Ausgang bzw. Eingang mit einer Nummer zu kennzeichnen.

Im Beispiel wird der Ausgangsport 1 des Elementes "IN_Taster" beschrieben, das in der dritten Hierarchieebene direkt zugehörig zu der "ECU_1" ist, die in der zweiten Hierarchieebene liegt.

<DISTANZ1>;<DISTANZ2>;...;<DISTANZn>

Die <DISTANZx>-Einträge bezeichnen den Grad der Abhängigkeit von <ELEMENT> von dem Element in der Tabellenzeile x. Für die Distanzen ist der Zahlenraum von 0 bis 126 zugelassen. Die Distanz 0 bedeutet, dass <ELEMENT> vollkommen unabhängig von dem Element in Tabellenzeile x ist.

Die Tabelle besteht aus insgesamt genau n Zeilen mit jeweils genau n Distanzeinträgen. Im Beispiel würde die Tabelle also aus 6 Zeilen bestehen.

Abbildungsverzeichnis

Tabellenverzeichnis

Lebenslauf

Persönliche Daten

Name	Mirko Harms
Geburtsdatum	10.4.1975 in Wolfenbüttel

Beruflicher Werdegang

2006 – heute	Entwicklungsingenieur bei der IAV GmbH, Gifhorn
2001 –.2005	wissenschaftlicher Mitarbeiter am Institut für Elektrische Messtechnik an der Technischen Universität Braunschweig
1999 – 2001	Hard- und Software-Entwickler für Supinfo, Braunschweig
1994 – 1995	Grundwehrdienst in Achim

Studium und Schulischer Werdegang

1995 – 2001	Studium der Elektrotechnik an der TU Braunschweig Vertiefungsrichtung Datentechnik
1985 – 1994	Gymnasium Bad Saulgau und Wolfenbüttel
1981 – 1985	Grundschule Schandelah und Bad Saulgau